

UNCLASSIFIED

DEFENCE RESEARCH ESTABLISHMENT
CENTRE DE RECHERCHES POUR LA DÉFENSE
VALCARTIER, QUÉBEC

DREV TM 1999-124
Unlimited Distribution / Distribution illimitée

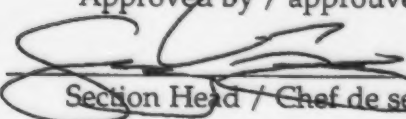
GIS PROTOTYPING WITH GENASYS PRODUCTS

by

G. Thibault and M. Lizotte

October / octobre 1999

Approved by / approuvé par


Section Head / Chef de section

14 10 99
Date

SANS CLASSIFICATION

RESTRICTIVE LEGEND

GOVERNMENT
OF CANADA

GOUVERNEMENT
DU CANADA

THIS DOCUMENT AND THE INFORMATION IT CONTAINS
IS DISCLOSED IN CONFIDENCE TO:

FOR DEFENCE INFORMATION PURPOSES ONLY, WITH NO
DISCLOSURES TO OTHERS OR ANY OTHER USE (INCLUDING
ANY MANUFACTURING USE) TO BE MADE WITHOUT THE
WRITTEN PERMISSION OF THE CANADIAN DEPARTMENT OF
NATIONAL DEFENCE (via DRDIM 3).
THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION.

WARNING NOTICE

The information contained herein is proprietary to Her Majesty and is provided to the recipient on the understanding that it will be used for information and evaluation purposes only. Any commercial use, including use for manufacture, is prohibited. Release to third parties of this publication or of information contained herein is prohibited without the prior written consent of DND Canada.

© Her Majesty the Queen in Right of Canada as represented by the Minister of National Defence,
1999

UNCLASSIFIED

i

ABSTRACT

Any automated Command and Control Information System (C2IS), in order to be successful, must rely on an efficient geographical information system (GIS). The purpose of this Technical Memorandum (TM) is to present the results of prototyping activities aimed at exploring one technological alternative with the Genasys family of products. It presents Raster/Vector display alternatives and military symbology alternative solutions. Many C++ scripted solutions, including the implementation rules, and the performance evaluations are provided. This study covers the R&D done during the Jan-Apr 1997 period.

RÉSUMÉ

Tout système automatisé de commandement et contrôle doit pouvoir s'appuyer sur un système de géomatique efficace pour avoir du succès. Le but de ce memo technique est de présenter les résultats des activités de prototypage explorant une alternative technologique avec la famille d'outils de géomatique Genasys. Le rapport présente des solutions alternatives pour la présentation Raster/Vecteur, et des symboles militaires sur les cartes. Plusieurs solutions sont explicitement données en langage de programmation C++, incluant les règles d'implémentation, et les résultats de performances observées lors des tests d'évaluation. Cette étude couvre des activités de R&D faites durant la période de janvier à avril 1997.

SANS CLASSIFICATION

TABLE OF CONTENTS

ABSTRACT/RÉSUMÉ.....	i
EXECUTIVE SUMMARY.....	v
1.0 INTRODUCTION	1
1.1 Overview.....	1
1.2 Conventions.....	3
1.3 Issues Identified in the Previous Studies	4
2.0 RASTER/VECTOR DISPLAY ALTERNATIVES.....	7
2.1 Goal.....	7
2.2 Result Summary.....	7
2.3 Approach Used.....	8
2.4 Recommended Approach.....	13
2.5 Implementation Rules (Annex B).....	15
3.0 IMPORT OF ADRG FILES FROM GEOMATICS CANADA CD ROM.....	16
3.1 Goal.....	16
3.2 Results Summary.....	16
3.3 Approach Used.....	16
3.4 Recommended Approach.....	18
3.5 Implemented Rules (Annex B).....	18
4.0 GIS TOOLBOX BUILDING.....	19
4.1 Goal.....	19
4.2 Results Summary.....	19
4.3 Approach Used.....	19
4.4 Recommended Approach.....	28
4.5 Implemented Rules (Annex B).....	29
5.0 SYMBOLOGY ALTERNATIVE SOLUTIONS.....	30
5.1 Goal.....	30
5.2 Results Summary.....	30
5.3 Approach Used.....	31
5.4 Recommended Approach.....	41
5.5 Implemented Rules	41
6.0 SIMULATE THE CLASS ENCAPSULATION OF RDBMS.....	42
6.1 Goal.....	42
6.2 Results Summary.....	42
6.3 Approach Used.....	42
6.4 Recommended Approach.....	44

UNCLASSIFIED

iii

6.5 Implemented Rules	44
7.0 INITIAL DISPLAY OF TACTICAL DATA.....	45
7.1 Goal.....	45
7.2 Results Summary	45
7.3 Approach Used.....	46
7.4 Recommended Approach.....	50
7.5 Implemented Rules (Annex B)	50
8.0 TOOLBOX - SELECT AND QUERY FUNCTIONALITY.....	51
8.1 Goal.....	51
8.2 Results Summary	51
8.3 Approach Used.....	51
8.4 Recommended Approach.....	53
8.5 Implementation Rules	53
9.0 TOOLBOX - PLACE UNIT SYMBOL FUNCTIONALITY.....	54
9.1 Goal.....	54
9.2 Results Summary	54
9.3 Approach Used.....	54
9.4 Recommended Approach.....	54
9.5 Implementation Rules	55
10.0 SITUATION MAP REFRESHES.....	56
11.0 COORDINATE MANAGEMENT (MAP PROJECTION, ETC.)	56
12.0 AREA SYMBOLS DISPLAY AND PLACEMENT	56
13.0 CONCLUSION.....	57
13.1 Context	57
13.2 Results	58
13.3 Final Note.....	59
14.0 REFERENCES	60
ANNEX A - THE INTERNAL STRUCTURE	A-1
ANNEX B - IMPLEMENTATION RULES.....	B-1
ANNEX C - PERFORMANCE & STATISTICS.....	C-1
FIGURES AND TABLES	

EXECUTIVE SUMMARY

Intelligence is an element of operations and an important component of decision-making. It enables commanders to successfully plan and conduct operations, identify targets, and win decisive battles. Ten years ago, the Defence Research Establishment Valcartier (DREV) began a research project to explore means to provide assistance to intelligence analysts responsible for the fusion of tactical information. This investigation led to the development of an experimental processing environment called the "Tactical Information Fusion" (TIF) prototype [1].

The purpose of the TIF prototype was to experiment with and provide tools to the All Source Analysis Team. The tools were experimented in order to improve decision-making, to accelerate registration and analysis of enemy activities, and to assist the Senior Analyst (SA) and the Duty Officer (DO) in the production of intelligence. There were two major field trials conducted in 1994 and 1995, plus some minor evaluation exercises that were conducted during the 1994-1997 period with the TIF prototype. The scope of the trials was voluntarily limited to the evaluation of the SA and the analyst functions, thus concentrating mostly on the processing phase of the Intelligence Cycle [2-4].

Other R&D activities were initiated to continue the enrichment of the corporate knowledge. DREV has designed and produced automated tools to support the development of a modern tactical intelligence processing and analysis capability for the Canadian Land Force. Under the DREV "Advanced Concepts for Tactical Information Fusion" (ACTIF) project many studies were conducted to assist in the formulation of a strategic plan for a tactical intelligence system [5] and a technological infrastructure architecture was

UNCLASSIFIED

v

developed [6] to support this plan. The design for a future automated tactical intelligence system is underway and is well documented [7-9]. The concepts, definitions and requirements for a future tactical intelligence system were investigated in a previous DREV study [10].

If there is one lesson to be learned from all the experience DREV has gained during those years conducting R&D in the command and control arena is the following one. In the future, for any automated Command and Control Information System (C2IS) to be successful, it will have to rely on an efficient geographical information system (GIS). Therefore, three studies were conducted to identify promising GIS technological alternatives for the ACTIF architecture [11-13]. The purpose of this Technical Memorandum (TM) is to present the results of prototyping activities aimed at exploring one technological alternative with the Genasys family of products. It presents Raster/Vector display alternatives and military symbology alternative solutions. Many C++ scripted solutions, including the implementation rules, and the performance evaluations are provided. This study covers the R&D done during the Jan-Apr 1997 period.

SANS CLASSIFICATION

1.0 INTRODUCTION

1.1 Overview

Ten years ago, the Defence Research Establishment Valcartier (DREV) began a research project to explore means to provide assistance to intelligence analysts responsible for the fusion of tactical information. This investigation led to the development of an experimental processing environment called the "Tactical Information Fusion" (TIF) prototype [1].

The purpose of the TIF prototype was to experiment with and provide tools to the All Source Analysis Team. The tools were experimented in order to improve decision-making, to accelerate registration and analysis of enemy activities, and to assist the Senior Analyst (SA) and the Duty Officer (DO) in the production of intelligence. There were two major field trials conducted in 1994 and 1995, plus some minor evaluation exercises that were conducted during the 1994-1997 period with the TIF prototype. The scope of the trials was voluntarily limited to the evaluation of the SA and the analyst functions, thus concentrating mostly on the processing phase of the Intelligence Cycle [2-4].

Other R&D activities were initiated to continue the enrichment of the corporate knowledge. DREV has designed and produced automated tools to support the development of a modern tactical intelligence processing and analysis capability for the Canadian Land Force. Under the DREV "Advanced Concepts for Tactical Information Fusion" (ACTIF) project many studies were conducted to assist in the formulation of a strategic plan for a tactical intelligence system [5] and a technological infrastructure architecture was developed [6] to support this plan. The design for a future automated tactical

intelligence system is underway and is documented [7-9]. The concepts, definitions and requirements for a future tactical intelligence system are given in a previous DREV study [10].

Any automated Command and Control Information System (C2IS), in order to be successful, must rely on an efficient geographical information system (GIS). Therefore, three studies were conducted to identify promising GIS technological alternatives for the ACTIF architecture [11-13]. The purpose of this Technical Memorandum (TM) is to present the results of prototyping activities aimed at exploring one technological alternative with the Genasys family of products.

The exploration was conducted following eleven (11) topics. These topics answer issues raised in two previous ACTIF GIS Studies [11-12] and other important issues related to the ACTIF architecture.

All results are documented in this report and some were also integrated into a common environment called the "Situation Map Prototype". This documentation has mainly two objectives:

- a) Determine how well Genamap meets the ACTIF architecture needs; and
- b) Document the know-how acquired during this prototyping activity in order to benefit from this learning if GenaMap¹ is finally selected for ACTIF.

The functional aspect was not really addressed by the prototyping team. When possible, the ACTIF functional approach was adopted but otherwise,

no time was taken to validate the functional approach. For that reason, rules that will be documented are technological i.e. related to the technical integration of the GenaMap products.

The first section summarizes the results concerning the issues identified in the first GIS study [11]. Each remaining section of the document is divided into the following subsections:

- .1 Goal;
- .2 Results Summary;
- .3 Approach Used;
- .4 Recommended Approach;
- .5 Implementation Rules (detailed in Annex B – Implementation Rules).

1.2 Conventions

This document follows specific typographic conventions, outlined below:

- Bold** is used for directories, headings, highlighting mandatory information in Makefile;
- Italic* is used to highlight source code examples, variable parameters;
- " is used to show Genasys family of product functions; and
- Text within frame indicates how issues were addressed within the current project.

¹ GenaMap is a registered trade mark of Genasys Co.

1.3 Issues identified in the previous studies

From the first GIS study [11], two limitations were pointed out: the display speed of raster data and the risk of a software crash associated with a complex setup.

1.3.1 Display Speed

This limitation was pointed out after ACTIF development team members had visited Fort Collins for the first GIS study [11] in March 1996. At that time, demonstrations were conducted using ADRG raster files translated into GenaMap type 7 files (GenaMap native raster format) and no specific tiling schemes were used. After some optimization of the display engine, the Genasys team was able to produce a faster display speed.

During the current project, two prototypes were developed using ADRG raster files. They demonstrated that the display refresh could be accomplished in under one second using the appropriate GenaMap module with some optimization.

Basically, optimization involves the following factors: the format and size of a resulting file, the number and size of tiles and the choice of display commands. The GenaMap family of products can handle a variety of raster formats, namely type 7, JPEG and compressed TIFF. It should be noted that JPEG and TIFF formats do not hold projection information. This information is saved in external files used by the display driver. JPEG and compressed TIFF formats produce smaller-sized files. This feature is useful as the display engine is sensitive to file size. On the other hand, using JPEG format systematically introduces losses of information while the compressed TIFF format does not.

All prototyping during the current project used the TIFF format with a tiling scheme. The ADRG files were imported and converted using the GenaMap utilities provided. The resulting «.IF» or «.FH» file held the projection information used by the display driver.

The large ADRG image is subdivided for purposes of display management into a set of smaller rectangles or tiles. The tiling scheme is very important for an optimal display speed.

Tests were conducted during the prototyping development in order to find the tiling scheme best suited to the ADRG files. The resulting TIFF file was tiled in a 256 x 256 scheme (65536 pixels per tile).

As a wide choice of commands is available in the GenaMap family of tools, display of raster files can be achieved in more than one way. GenaVive and GenaCell² modules can display raster data. Inside these modules, a couple of sets of commands can be used ('pixshade', 'gvplot', etc.).

The optimum choice of command, used through the second prototype, was to combine the GenaVive module with the Displayer module. Using an imported ADRG file, we have been able to «cache» a 25 Mb tiled TIFF file using Displayer functions and display it using GenaVive commands in under one second.

1.3.2 Risk of software crash

The GenaMap Application Program Interface (API) consists of two different parts, the Spatial API and the dispatcher («Request Broker» API). For the ACTIF development, it is suggested that the two API components be used. For the last two years, Genasys has adopted a common «Request Broker» architecture for their new modules (Displayer, GenaVive, etc.). The «Request Broker» API is built on the same architecture. In order to have the

² GenaVive and GenaCell are part of the GenaMap family of products from Genasys Co.

best available range of functionality, the «Request Broker» API must be used. However, for accessing basic commands in modules such as GenaMap and GenaCell, it is more suitable to use the Spatial API than the «Request Broker» API because the Spatial API calls only the required portion of the modules.

Tests have been conducted during the prototyping development in order to integrate the Spatial API and «Request Broker» API. The results showed that mixing «Spatial API» and «Dispatcher API» could lead to problems if we were not careful. We discovered that attaching GenaMap in the initialization script and removing "Command Encapsulation" functions like: 'GExec', 'GSession', 'GExit', and 'GFetchMessage' was the correct way to proceed. Thus we implemented the first prototype as a «Service Requester» to the dispatcher and integrated Spatial API code as well without experiencing a software crash. All demonstrations held at DREV used the «Request Broker» architecture in a synchronous mode of processing. No problems were found using the «Request Broker» API and the Spatial API in that way. We did not implement any prototype with distributed processes across networks.

The research and development activities described in this document were performed at DREV between April 1997 and March 1998 under DRDB Thrust Number 2JB11 Katimavik and 2JB13 Advance Concepts on Tactical Information Fusion (now changed to 2KL13 and 2KL15 under the new thrust numbering system).

2.0 RASTER/VECTOR DISPLAY ALTERNATIVES

2.1 Goal

It is essential for a good command and control information system (C2IS) using a GIS to possess the capability of displaying tactical information superimposed on digital maps. Therefore in this context, it is important to explore the capability of a GIS to exploit both raster and vector data alternatives. In order to study possibilities related to raster and vector maps, four setups have been successively tested.

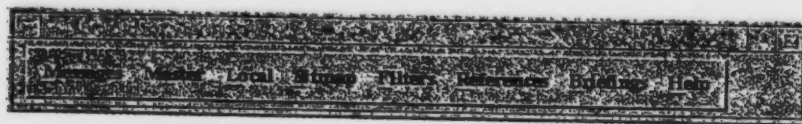


FIGURE 2.1- Int Manager prototype

- a) Encapsulation of the Genasys session in the «IntManager» prototype and display of a vector type map;
- b) Encapsulation of the Genasys «Request Broker» architecture in the «IntManager» prototype and display of a vector type map;
- c) Encapsulation of the Genasys «Request Broker» architecture in the «IntManager» prototype and display of a raster type map; and
- d) Combining both vector and raster type in the same application.

2.2 Results Summary

The integration of both raster and vector data has been achieved during the prototype. Nevertheless, a better level of integration can be achieved. This prototype revealed no problems implementing a mixed raster and vector solution. A more elegant way to implement graphics windows into the main application would be to use native X11 windows generated directly by UIM/X. This will probably be the path to follow in the ACTIF development.

This prototype also showed that GenaVive service alone is not fast enough to provide the targeted display speeds. Displayer will also be required to achieve an application that would meet user's expectations in terms of display speed.

Finally, this prototype provides the development team many ways to implement various ACTIF display modules, but also clarifies what to put in place for development purposes.

2.3 Approach Used

In order to correctly connect to a GenaMap session we must include a 'GSession(session_name)' call to our main application. This step was handled by the constructor of IntManager class. The 'GSession' function creates the files «WORK.PN», «INTERPRO» and all the required directory in the current work area. Moreover, our session_name is created within the ZF20 directory. The 'GExit' function, responsible for the session termination process, is included in the destructor clause of the IntManager class object. One way to draw a map within a window is to use the GenaMap function 'GExec("plot map");'. This kind of implementation is called: «Command Encapsulation». It returns output messages in a message loop accessible via the 'GFetchMessage' and 'GSprintMsg'.

Following the «Request Broker» architecture, we did adapt the IntManager prototype to become a «Service Requester». We registered two callback functions to process the standard output and the error output of the dispatcher via the global variable «io». We then went active with Dispatcher using 'ProductActive(TRUE)'. The results are shown in `actifPrototype/CommonFeatures/Application/src/intManager.cpp`:

```
{  
  SERV_ID io[3];
```

UNCLASSIFIED

9

```

int OutputIt(SERV_ID, SERIAL *);
io[SERV_OUT] = ServiceProvide(SERV_ANONYMOUS, NULL,
    OutputIt);
io[SERV_ERR] = ServiceProvide(SERV_ANONYMOUS, NULL,
    OutputIt);
...
    /*-----
    * Go Active with dispatcher
    *-----*/
    if( ProductActive(TRUE) )
    {
        fprintf(stderr, «Failure to activate the
            interface response channel\n»);
    }
    else
    {
        fprintf(stderr, «\nStarting
            dispatcher...\n»);
    }
...
ProductActive(FALSE);
    fprintf(stderr, «\nClosing dispatcher...\n»);
    DispShutdown();
}

int OutputIt(SERV_ID service, SERIAL *s)
{
    char *str = NULL;

    UnpackString(&str, s, SER_DECODE);
    if (service == io[SERV_OUT])
        fprintf(stderr, «Data: %s\n», str);
    else
        fprintf(stderr, «ERROR: %s\n», str);
    UnpackString(&str, s, SER_FREE);
    DispExit(0);

    return(0);
}

```

This piece of code is executed when the dispatcher starts and attaches the intManager process. For more transparency we used an alias (see Annex A - The inside stuff). You can then send commands to peer clients via 'PackArgs' and 'DispExec' (ex: send the string «plot map» to client gmap); wait for reply using 'DispWait' and free memory that PackArgs allocated. When

SANS CLASSIFICATION

finished, shut down the dispatcher and the intManager client process using 'ProductActive(FALSE)' and 'DispShutdown' clauses.

Plotting a map using any Genasys client (i.e., calling the «Request Broker» explicitly) is similar to plotting a map with GenaMap on the command line (i.e., without calling the «Request Broker» explicitly). The difference is handled by passing the appropriate service number to the 'DispExec' function.

ex: DispExec(Service number, &message, NULL);

TABLE 2.1: SERVICE NAME AND NUMBER

Service Name	Service number
GenaVive	SERV_GENAVIVE
GenaMap (gmap)	SERV_GENAMAP
Displayer (Xfer)	SERV_XFER
Scripting (Dispscript)	SERV_SCRIPT
Transient service	SERV_ANONYMOUS

There are three possible avenues when drawing a raster map within a window: GenaCell, GenaVive and Displayer. Their differences are mainly in display speed and caching possibilities.

Outputting raster and vector maps in the same X11 window causes some difficulties. Different kinds of draw able functions are available with the Genasys product. They are outlined in Table 2.3.

TABLE 2.2: POSSIBLE AVENUES FOR DISPLAYING RASTER MAPS

GenaCell	This service is used when manipulating the raster file data for computational purposes. The display speed does not meet our requirement because there is no caching of the image being read. Thus refreshing became unacceptable. For those reasons, none of the prototypes used GenaCell.
GenaVive	This service implements a «cache» that lets us rapidly display the raster map (or any overlays). It manages the map more efficiently using a tiled or stripped rows scheme. The prototype Demo 1 used the GenaVive service.
Xfer (Displayer)	This service is dedicated for display purposes. It lets us manage different kinds of layers such as: raster, vector or glyph. Cached image instances allow us instant refreshing and fast update of any layers. The prototype Demo 2 showed the Displayer capabilities.

TABLE 2.3: DIFFERENT KINDS OF DRAW ABLE FUNCTIONS

Command Line	API	Map Types
! output 3 window or ! wcreate window	'GDWCreate' or 'GDWCreateInteractive'	Vector map (type 4)
! gvcreate window	'DispExec(SERV_GENA VIVE, "gvcreate window")'	Vector map (type 4) Raster map (type 7 and «.tif»)
! gw cr im window or ! gw cr -stdcmap default window	'DispExec(SERV_SCRIP T, "gxwcreate -geometry width x height + x + y - stdcmap default imaging window")'	Raster map (type 7 and «.tif»)

The first approach in Table 2.3 cannot output raster files and the two others have some inherent differences in their behaviour. They handle the expose event and "backingstore" properties differently. The approach used was to create a draw able window using the third approach and retrieve its

X11 window identification called «xid». The prototype draw able window supported exposure but did not completely implement the "backingstore" property when minimized. It was implemented like this:

```
// Create an X Drawable Window and gets its xid.
sprintf(command, «gw cr -geometry %dx%d+%d+%d
-stdcmap default imaging %s», WIDTH, HEIGHT,
POSX, POSY, windowName);
execute_dispatch_command(SERV_SCRIPT, command);

sprintf(command, «xwininfo -name %s >%s»,
windowName, temp);
system(command);

if((fp = fopen(temp, «r»)) == NULL)
{
    fprintf(stderr, «Unable to retrieve
window id.\n»);
    return(1);
}

fscanf(fp, «\n%21c%x», command, &xid) ;
fclose(fp) ;
remove(temp);
```

Displaying raster and vector files may lead to chaos if we mix «Spatial API» and «Dispatcher API». We encapsulated the 'GExec' and 'DispExec' using two distinct methods shown in Table 2.4.

TABLE 2.4: ENCAPSULATION METHODS

Spatial API	Dispatcher API
<pre>BOOLEAN execute_genamap_command (char * command // The command line) { if (!GExec(command)) { fprintf (stderr, «Error from GExec():\n%d %s\n», GErrorNum(), GerrorMess()); } }</pre>	<pre>BOOLEAN execute_dispatch_command (SERV_ID service // Refer to Table 1.2 char * command // The command line) { DISP_ID req; int stat = 0; extern SERV_ID io[3]; SERIAL message;</pre>

```

        return(FALSE);
    }
    return( TRUE );
}

// Insert here a command to
// split the command line
// into argc argv
PackArgs(argc, argv, io,
        &message);

req = DispExec( service,
        &message, NULL);
if( req < 0 )
{
    if( CheckDispatcherEvent(0)
        && GotDispatcherEvent()
        == DISP_CONN_LOST )
        fprintf( stderr,
            «\nConnection lost to
            the intManager\n» );
    else
        fprintf( stderr, «\n Could
            not submit command to
            intManager\n» );
}
else
while( DispWait(&stat) !=
    req )
    fprintf(stderr,
        «\nWaiting...\n»);
if( stat) fprintf( stderr,
    «\nGenaVive didn't like
    the command\n»);
}
free((void *) message.data);
free(argv);
return(TRUE);
};

```

2.4 Recommended Approach

As we mentioned previously, mixing «Spatial API» and «Dispatcher API» may lead to chaos if we do not take care. In future development, we suggest attaching the 'gmap' client to the «Request Broker» in the initialization script and removing «Command Encapsulation» functions such as: 'GExec', 'GSession', 'GExit', 'GFetchMessage', etc. This approach («Command Encapsulation») would be replaced by passing the GenaMap

commands through the dispatcher 'DispExec' function using either the scripting client (Dispscript) or the GenaMap client ('gmap').

At the time we wrote this document, it was clear that the ACTIF IntManager application would become a «Service Requester» module built for the Genamap «Request Broker» architecture. In that way, the current prototype implementation will remain.

Based on, at the time, current evaluation of Genamap (version 6.2) services, it is clear that Displayer and GenaVive should be implemented to manage all graphical display in the ACTIF GIS development. Thus the knowledge accumulated in section 2.3 will remain useful unless the new version of Genamap (version 7.1) offers a new and better alternative.

As we saw in section 2.3, many scenarios exist when managing an X11 window. There is one possibility explored during the construction of the prototype but not implemented nor documented in the above-mentioned section: UIM/X. This graphical user interface builder let us manage window event callback more efficiently. The reason why no documentation has been produced is that it did not work at the time. The problem stems from the need to attach a standard colour map property to our UIM/X window. The customer support led us to some UNIX utilities available in the \$(GENA)/bin directory to manage colour map properties as shown in Table 2.5.

TABLE 2.5: MANAGEMENT OF COLOUR MAP PROPERTIES

Gxlistprop	Will list if a standard colour map property is available.
Gxaddcmap -stdcmap default -id xid	Will associate a standard colour map property to an x11 graphics window known by its xid

The idea is to create a graphics window via UTM/X and issue the «gxaddcmap» on that window.

Note: If you issue this command through the dispatcher to Dispscript, you will need to append a «sys» clause before the command since it is a UNIX command.

Man pages are available in \$(GENA)/man/cat1 for those functions. This latest comprehension of the problems encountered leads us to believe that it is this approach that should be implemented in future development.

2.5 Implementation Rules (Annex B)

The reader will find the implementation rules and the necessary scripted routines at Annex B to perform the following functions:

- a) Connecting to a Genasys Session; and
- b) Executing Request broker commands.

3.0 IMPORT OF ADRG FILES FROM THE GEOMATICS CANADA CD-ROM

3.1 Goal

It is essential for a good command and control information system (C2IS) using a GIS to possess the capacity to read map files directly from CD-ROM. Therefore in this context, it is important to explore the capability of a GIS to manage ADRG raster data. In order to test the ADRG translator, the team managed to import ADRG files and tried to find the best combination of tiling scheme and file format. The primary intention was to reduce the size and optimize the display speed of raster files. Because ADRG files are huge, an alternative file format had to be used. This section addressed also the capacity to read the files directly from CD-ROM and the size that such files required on disk.

3.2 Results Summary

Direct reading from CD-ROM is not feasible because ADRG files cannot be directly read as input for GenaVive. First, ADRG files must be converted into Genamap raster files (type 7) and then translated from type 7 to compressed TIFF in order to get better speed performance results with GenaVive. During the prototype development, a quicker way to reference a TIFF file was found. Rather than use the GenaVive registration utility provided by Genasys, simply rename the file «*.FH» into «*.IF» generated by the translating process.

3.3 Approach Used

The import process consists of the three steps presented in the following sub-sections.

3.3.1 Convert the ADRG file into a GenaMap type 7

The following command was executed on the UNIX command line; it could have been implemented in a graphical user interface as well:

```
$(GENA)/map_6.2/contrib/bin/adrg2gc -path
    cdrom_path_to_transh01.thf_directory -interpolate -
    colortable - -verbose
```

We are prompted for each raster file_name on the Geomatics Canada CD-ROM. The type 7 files and color table adrg.CT are created in the current directory. We can then copy the type 7 files to the target *workarea/ZF07/* directory and copy the color table adrg.CT to the target *workarea/ZF31/*

Statistics

An ADRG file xxca0103.img of ~213 Mb was converted into a type 7 file of ~93 Mb called xxca0103.RD in approximately 45 minutes on a Sparc station 5. To provide a useful point of reference, the geographic coverage of xxca0103.img was equal to three map sheets of 150km by 100km digitized at 1:250k of the Vancouver Island, Canada.

3.3.2 Optimize the GenaMap type 7 into a TIFF tiled file

The following command was executed on the UNIX command line; it could have been implemented using the Dispatcher architecture as well:

```
$(GENA)/bin/gvconvert -type tiff -format tiled 256 256 -
    colortable adrg.CT file_name.RD file_name.tif
```

It may take some time to find the best tiling ratio. Usually 256 x 256 pixels gives a smaller TIFF file. The «.tif» file is created in the current directory. We can then copy the ".tif" and «.FH» (equivalent of the «.IF» file) to the target *workarea/raster/* directory.

Statistics

A type 7 file xxca0103.RD of ~93 Mb was optimized into a file of ~24 Mb called xxca0103.tif in approximately 5 minutes on a Sparc station 5.

Note: The information thrown away during the conversion process is most likely to be the color palette and any "meta-data" associated with a 24 bit format like ADRG. These losses are not significant for a command and control application like ACTIF because ACTIF requires a geo-referenced background image only.

3.3.3 Create the «IF» file

Instead of replacing the information file by the «.FH» file, we could also have used the GenaVive registration utility to create the corresponding «.IF» file. The following command was executed on the UNIX command line:

```
$ (GENA)/bin/register workarea session_name.
```

Then, follow the GenaVive Registration Utility User Manual.

3.4 Recommended Approach

No further analysis was required within the prototype development. The «import file» process could be implemented within the «Request Broker» architecture as another ACTIF «Service Requester» module.

3.5 Implementation Rules (Annex B)

The reader will find the implementation rules and the necessary scripted routines at Annex B to perform the function: Converting ADRG.

4.0 GIS TOOLBOX BUILDING

4.1 Goal

It is essential for a good command and control information system (C2IS) using a GIS to possess a high level of interoperability amongst its components. Therefore in this context, it is important to explore how easy it is to apply a middleware architecture to a GIS. This section is dedicated to the integration of UIM/X graphical user interface windows with Genasys graphic windows. The intent is to find ways to implement the «Request Broker» architecture in the main application. This section benefits from the knowledge acquired in section 2.

4.2 Results Summary

Two prototypes have been developed to illustrate different components. The first one, illustrating the main features (panning, zooming, querying distance and querying objects), is detailed in the next sub-section. The second one, more specific to the Displayer approach, focused on the possibilities of that service and the display speed. The capability to achieve a fast display resides in the ability to «cache» in memory a large portion of a file shown in a window. For that reason, files have to be «cached» entirely once for each required scale value prior to use.

4.3 Approach Used

We created two independent toolboxes illustrating our two distinct prototype approaches. The toolbox of Demo 1 demonstrates six (6) features linked to the work of a «Senior Analyst». The main issue is well handled for vector maps using GenaMap vector functions and is more tricky for raster maps: the MBR (minimum bounding rectangle i.e. north-west coordinate and south-east coordinate) must be re-calculated (before the raster map is

redrawn) each time the vector view changes. We used the 'getv wind' function to read the MBR values. We retrieve the corresponding values from the message queue using 'GFetchMessage' and 'GSprintMsg'. The drawing function 'gvdraw' is then called with the new calculated MBR values. Table 4.1 describes the toolbox function.

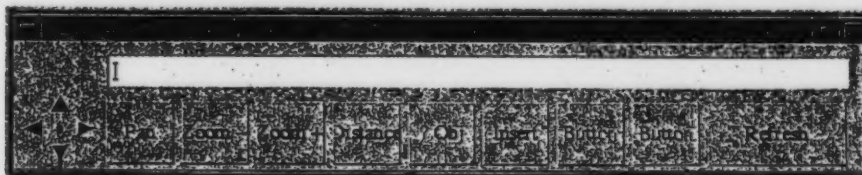


FIGURE 4.1- Demo 1 toolbox

TABLE 4.1: DEMO 1 TOOLBOX FUNCTION

Toolbox button	Description
Pan	This ability allows the user to re-center the map to a specific point.
Zoom -	This ability allows the user to display the previous map scale at a specific location (4 levels are pre-defined).
Zoom +	This ability allows the user to display the next map scale at a specific location (4 levels are pre-defined).
Distance	This ability allows the user to calculate the distance between two or more points on a map.
? Obj	This ability allows the user to interrogate and update any symbol (unit) on a map.
Insert	This ability allows the user to insert a symbol (unit) on a map.

4.3.1 Pan

An offset is calculated from the user click position on the vector map and passed to the scroll function. The calculated MBR is passed to 'Gvdraw'.

```

void WorkingSitMapView::vPanMap()
{
    char command[255];
    double x1,x2,y1,y2;
    BOOLEAN status;
    MMESSAGE *message;
    char msgContainer[2048];
    execute_genamap_command («scroll;getv wind; er»);
    for (int i=0; (status=GFetchMessage( &message )) &&
        message != NULL;i++)
    {
        if( GSprintfMsg(msgContainer, message) > 0)
        {
            switch(i)
            {
                case 1: sscanf(msgContainer, «%lf»,
                    &x1);break;
                case 2: sscanf(msgContainer, «%lf»,
                    &x2);break;
                case 3: sscanf(msgContainer, «%lf»,
                    &y1);break;
                case 4: sscanf(msgContainer, «%lf»,
                    &y2);break;
                default: break;
            }
        }
        else
        {
            fprintf(stderr,«Could not access
            message.\n»);
            return;
        }
    }
    if( !status ) {
        printf («Error from GFetchMess():\n%d
        %s\n», GErrorNum(), GErrorMess() );
        return;
    }
    sprintf(command, «gvdraw WS1 WS1 default %lf
    %lf %lf %lf mapWS1», x1,y1 ,x2,y2);
    execute_dispatch_command(SERV_GENAVIVE,
    command);
    execute_genamap_command («plot cotel bu 260
    nob»);
};

```

Statistics

Depending on the map scale, the performance of the panning function varies. The variability in refresh time is greater at higher map scales. This is related to the fact that the density of data points is higher at higher map scales. When the panning operation forces new data to be loaded into memory from disk, transfer times can be much longer at higher map scales due to the larger data volumes involved as shown in Table 4.2.

TABLE 4.2: REFRESH TIME

Scale	Time to refresh
1: 1 000 000	5 to 35 seconds
1: 500 000	6 to 12 seconds
1: 250 000	6 to 8 seconds
1: 100 000	5 to 6 seconds

4.3.2 Zoom- / Zoom+

A new window scale is calculated at the user click position on the vector map. The calculated MBR is passed to 'gvdraw'.

TABLE 4.3: ZOOM FACTORS & SCALES

The variable zoomFactor stores the pre-defined window scale:	Scale
ZoomFactor[0]	1: 1 000 000
ZoomFactor[1]	1: 500 000
ZoomFactor[2]	1: 250 000
zoomFactor[3]	1: 100 000

UNCLASSIFIED

23

```

void WorkingSitMapView::vZoomInMap()
{
    char command[255];
    double x1,x2,y1,y2;
    BOOLEAN status;
    MMESSAGE *message;
    char msgContainer[2048];

    if( currentZoom >= 3)
    {
        currentZoom = 3;
    }
    else
    {
        sprintf(command, «zoom s %d ;getv wind»,
            zoomFactor[++currentZoom]);
        execute_genamap_command(command);
        for (int i=0; (status=GFetchMessage( &message
            )) && message != N ULL;i++)
        {
            if( GSprintMsg(msgContainer, message) > 0)
            {
                switch(i)
                {
                    case 1: sscanf(msgContainer, «%lf»,
                        &x1);break;
                    case 2: sscanf(msgContainer, «%lf»,
                        &x2);break;
                    case 3: sscanf(msgContainer, «%lf»,
                        &y1);break;
                    case 4: sscanf(msgContainer, «%lf»,
                        &y2);break;
                    default: break;
                }
            }
            else
            {
                fprintf(stderr,«Could not access message.\n»);
                return;
            }
        }
        if( !status ) {
            printf (»Error from GFetchMess():\n%d  %s\n»,
                GErrorNum(), GErrorMess() );
            return;
        }
        sprintf(command, «gvdraw WS1 WS1 default %lf %lf
            %lf %lf mapWS1», x1,y1,x2,y2);
        execute_dispatch_command(SERV_GENAVIVE, command);
        execute_genamap_command(«plot cotel bu 260 nob»);
    }
};

```

SANS CLASSIFICATION

Statistics

Depending on the map scale, the performances of the zooming function vary as shown in Table 4.4:

TABLE 4.4: REFRESH TIME

Scale	Time to refresh	
	Zoom-	Zoom+
1: 1 000 000	~23 seconds	
1: 500 000	~8 seconds	~7 seconds
1: 250 000	~6 seconds	~6 seconds
1: 100 000		~5 seconds

4.3.3 Distance

The distance is calculated between the first and the last point location. The user defines those point locations by clicking the left mouse button on the map. When finished, the user clicks the right mouse button. The distance is displayed in the toolbox text field using the session unit measurement and projection.

```
void WorkingSitMapView::vEvaluateDistance()
{
    BOOLEAN status;
    MMESSAGE *message;
    char msgContainer[2048];
    execute_genamap_command("distance");
    for (int i=0; (status=GFetchMessage( &message )) &&
        message != NULL;i++)
    {
        if( GSprintfMsg(msgContainer, message) > 0)
        {
            switch(i)
            {
                case 1: UxPutText(toolBox->textField1,
                    msgContainer);break;
                default: break;
            }
        }
    }
}
```

UNCLASSIFIED

25

```
    }  
    }  
    else  
    {  
        fprintf(stderr, «Could not access  
message.\n»);  
        return;  
    }  
}  
if( !status )  
{  
    printf («Error from GFetchMess():\n%d  
%s\n»,  
    GErrorNum(), GErrorMessage() );  
    return;  
}  
};
```

4.3.4 ?Obj

The Query object process is detailed in Section 8.0 "Toolbox - Select and Query Functionality".

```
void WorkingSitMapView::vQueryObject()  
{  
    MyObject *anObject;  
    int unitId=0;  
    BOOLEAN status;  
    MMESSAGE *message;  
    char msgContainer[2048];  
    execute_genamap_command(«query tag»);  
    for (int i=0; (status=GFetchMessage( &message )) &&  
        message != NULL; i++)  
    {  
        if( GSprintMsg(msgContainer, message) > 0)  
        {  
            switch(i)  
            {  
                case 1: sscanf(msgContainer, "%d",  
                    &unitId); break;  
                default: break;  
            }  
        }  
        else  
        {  
            fprintf(stderr, «Could not access  
message.\n»);  
            return;  
        }  
    }  
    if( !status ) {
```

SANS CLASSIFICATION

```

printf («Error from GFetchMess():\n%d  %s\n»,
    GErrorNum(), GErrorMess() );
return;
}
if( unitId )
{
    anObject = new MyObject(unitId);
    anObject->vBuildWindw();
    anObject->gui->show();
}
};

```

4.3.5 Insert

The insert process is detail in Section 8.0 "Toolbox - Select and Query Functionality".

```

void WorkingSitMapView::vInsertObject()
{
    MyObject *anObject;

    execute_genamap_command («edmap cotel; gloc;
        edbn;»);

    anObject = new MyObject(this);
    anObject->vBuildWindw();
    anObject->gui->show();
};

```

The toolbox of Demo 2 demonstrates two (2) features linked to the work of a «Senior Analyst» where the focus is on display speed. This time, instead of recalculating the MBR before each drawing, the images are «cached» using Genasys Displayer client. The toolbox manages the visibility and the position of a «virtual window» over the «cache». The following Table 4.5 describes the toolbox function.

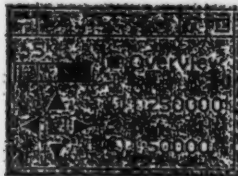


FIGURE 4.2- Demo 2 toolbox

TABLE 4.5: DEMO 2 TOOLBOX FUNCTION

Toolbox button	Description
Black arrows	This ability allows the user to scroll on the map view.
Radio button	This ability allows the user to choose the map scale (three levels pre-defined)
Horizontal scale	This ability allows the user to select an offset when scrolling.

4.3.6 Black arrows

The user clicks any arrow to move the «virtual window» over the «cache». Depending on the selected arrow, the function `vScrollInstance(heading)` is called. The heading parameters are 3-right, 6-bottom, 9-left and 12-up.

```

void IntManager::vScroll(int x, int y)
{
    char command[255];
    sprintf(command, «xfscroll 0x%x %d %d», xid, x, y);
    fprintf(stderr, «%s\n», command);
    execute_dispatch_command(SERV_XFER, command);
};

void IntManager::vScrollInstance(int heading)
{
    int offset = UxGetValue((_UxCuiIntManager*)gui)-
>scaleH1);

    switch(heading) {
    case 3:  vScroll(-offset,0); break;
    case 6:  vScroll(0,-offset); break;
    case 9:  vScroll(offset,0); break;
    case 12: vScroll(0,offset); break;
    default: ;
    };
};

```

Statistics

Using Displayer, each click on an arrow button produces an instant refresh of the «virtual window» over the «cache».

4.3.7 Radio button

The user clicks any radio button to change the map scale. Depending on the selected button, the Boolean visibility factor of the corresponding layer within the «virtual window» is set to true or false. This sample code is executed when the Overview button is clicked.

```
void IntManager::vOverview()
{
char command[255];
    sprintf(command, «xfivis %s %s 3», «Overview»,
«true»);
    execute_dispatch_command(SERV_XFER, command);

    sprintf(command, «xfivis %s %s 2», «250k»,
«false»);
    execute_dispatch_command(SERV_XFER, command);

    sprintf(command, «xfivis %s %s 1», «50k», «false»);
    execute_dispatch_command(SERV_XFER, command);
};
```

Statistics

Using Displayer, each click on a radio button produces an instant refresh of the «virtual window» over the «cache» at the new map scale.

4.3.8 Horizontal scale

The user slides the horizontal scale to increment or decrement the offset scroll.

4.4 Recommended Approach

The synchronization between raster and vector using MBR calculation is very time consuming. At the time we designed the prototype, another synchronization function between raster and vector was available: 'gvsync'. This function was not used because it did not work properly in our prototype architecture. In the target approach of section 2.3, we decided to remove all «Command Encapsulation» functions enabling us to bypass this particular problem. Thus, we would rather use 'gvsync', 'gvpan', 'gvzoom' functions in

UNCLASSIFIED

29

our future development if we do not need MBR values. Otherwise, a specific Object class will handle MBR values. At the time we wrote this document, our preferred approach was to use Displayer, GenaVive and Dispscript modules.

4.5 Implementation Rules (Annex B)

There are no rules to report for this function.

SANS CLASSIFICATION

5.0 SYMBOLOGY ALTERNATIVE SOLUTIONS

5.1 Goal

It is essential for a good command and control information system (C2IS) using a GIS to possess the capacity to use military symbology in accordance to specific military standards. Therefore in this context, it is important to explore the available approaches to represent military symbols with the Genamap family of products. The following capabilities must be provided in the implemented solution:

- a) Creation of symbol (multicolour);
- b) Display of text fields attached to symbols;
- c) Establishment of a library of symbols;
- d) Manipulation of symbol primitives; and
- e) Changing scale, visibility and appearance of symbols.

Two different approaches have been tested: the Genamap 6.2 symbology functions and the new symbology service (module) from Genamap 7.1 [14].

5.2 Result Summary

The symbology supported under GenaMap 6.2 does not meet every required item that were identified by the prototype development team. In particular, the incapability to support multicolour and attachment of multiple text fields to symbols.

Note: These features can be implemented under GenaMap 6.2 but not easily.

The new service «Graphics», available with GenaMap 7.1, is a more powerful solution and seems to offer everything that is required for ACTIF. There are still minor unanswered questions on integration with Displayer

that could not be addressed during the time and budget allowed for the prototype. Nevertheless, as these issues are considered minor, no doubts are raised on the feasibility of the solution.

5.3 Approach Used

This section is divided into two major sub-sections: Genasys 6.2 release versus Genasys 7.1 beta 3. The symbology management capabilities of both versions have been examined. The ACTIF system must handle multiple colour, text and basic symbols: marker, line or area. The following figure was taken from a military symbology document [15].

It represents the rules, character length and positioning of each piece of information as shown on Figure 5.1 (fields B, C, D, E, F, G, H, J, K, L, M, N, P, Q, R, S, T, V & W) around a basic symbol (field A). See the documentation for complementary information.

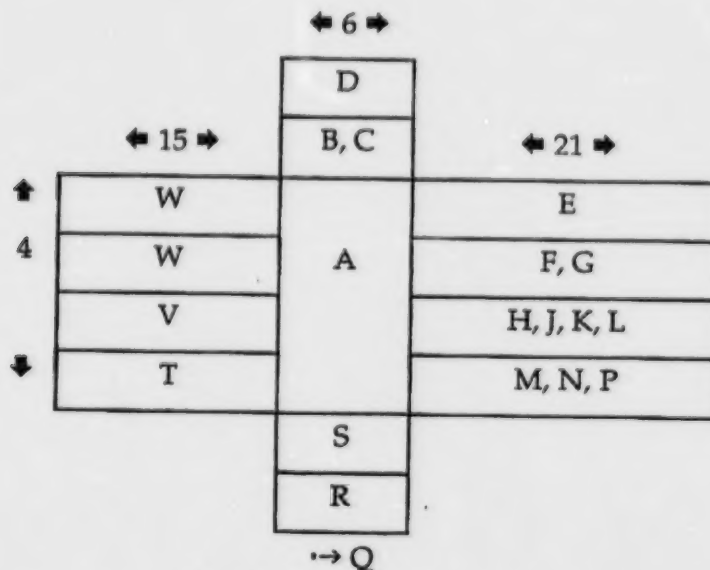


FIGURE 5.1- Military symbology fields

5.3.1 GenaMap 6.2

Using the current GenaMap version we did try some command line instructions to evaluate whether the symbology support could be used to provide the capabilities listed in Section 5.1.

The symbology in GenaMap 6.2 is divided in two distinct ways: structure and display. The font defines the structure of a symbol. Symbol font creation and maintenance are handled by the function 'font'. Each font is identified by a unique number and may have up to 128 point symbols, letters, or a combination of both. A Bundle table defines the symbology display parameters like: font, index, thickness, colour, width, height and rotation. The font and index number for markers, lines and fills must be taken from a font database. A Bundle table is created using function 'buncreate' and stored in the *workarea* /ZF30 directory. Symbols are made of only one colour. Multicoloured symbols can only be simulated by creating a set of single-colour symbols of appropriate dimensions and superposing them on the display.

Note: There is a limitation of one Bundle table open at a time in GenaMap 6.2 using the 'bundle' function. This is not the case in GenaMap 7.1.

Symbology entries are created and stored in a Bundle table using the function 'symbology'. An entry is referenced by its user-assigned index number and is one of the following four types:

- a) text (annotation);
- b) line styles (line features);
- c) markers (point features); or
- d) fill patterns (area features).

Text information is handled by what is called a GenaMap text map. A text map is stored in the *workarea* /ZF10 directory. Using function 'text', text annotation is created and assigned at a specific location.

A font database is a collection of fonts establishing a «library». GenaMap uses a font database called FONTS. A font database provides room for up to 128 fonts. Thus all military symbols required for the ACTIF system must be defined in font file stored in a font database before their access.

The 'assign' function links map features with symbology "id" stored in the current open Bundle table.

Once built and assigned, symbols are available for display on a situation map. In Section 8 we will discuss in detail the initial display process using symbol number 260 from Bundle table gmap.BT.

Parameters vary depending on the selected symbology. Using the 'symbology' function allows one to scale and size symbols. By changing the current Bundle table with 'assign', it is possible to modify the appearance of any symbols (using the same index number) on a map.

Note: We did not have time to look at the «glyph» approach from the Displayer [16-17] service in this section within the time frame of the prototype development. Nevertheless, this module offers significant advantages like event management of the display portion covered by the symbol (e.g. drag and drop, resizing, etc.), collision detection and smart refreshes (no need to refresh the entire display when you remove or move a symbol).

5.3.2 GenaMap 7.1b3

Using the new Graphics service provided by Genamap version 7.1 b3, we made some GenaMap scripts to demonstrate the available options for managing ACTIF symbology.

We created some procedures called «primitives» to construct our own military basic symbols library. Those procedures are based on «Parametric symbology» [14]. The idea is to build a symbol like a marker, line or area (see Figure 5.2) by grouping a number of primitives related to a context. This context is defined by the user input or the data itself.

```
#-----
#
# File:          military.symbols
# Description:   This file contains a test base for
#               military symbols used by the Genasys
#               family of product.
# Creation Date: This file was created by FAUDET on
#               April 97
# Last Modification: none.
#
#
#-----
```



FIGURE 5.2

```
#=====
#
# Procedure linked to field «A» basic symbol of the
# Military Symbols APP-6
#
#-----
#
# Coloured container
#
#
```



FIGURE 5.3

```
procedure unit marker ( number @r number @g number @b )
  #Solid Line
  colour ( RGBi ( @r @g @b ) )
  linestyle ( 0 0 0 ( ) )
  path ( ( -11 -6 -11 9 11 9 11 -6 close ) )
  fill ( )
endprocedure
```

```

#
# Formation or Unit: is a rectangle
#
procedure _unit marker ( number @r number @g number @b )
  #Solid Line
  path ( ( 0 0 ) )
  colour ( RGBi ( @r @g @b ) )
  linestyle ( 0 0 0 ( ) )

```

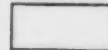


FIGURE 5.4

```

  linewidth ( 0.3 )
  path ( ( -10 -5 -10 5 10 5 10 -5 close ) )
  edge ( )
endprocedure

```

```

#
#
#
#
# A coloured rectangle environment: Local, Master
#
#

```



FIGURE 5.5

```

#
procedure environment marker ( number @r number @g
  number @b )
  #Solid Line
  colour ( RGBi ( @r @g @b ) )
  linestyle ( 0 0 0 ( ) )
  path ( ( -10 6 -10 9 10 9 10 6 close ) )
  fill ( )
endprocedure

```

```

#
# Identification: Enemy, Friendly, Neutral
#
#
#
#

```



FIGURE 5.6

```

procedure identify marker ( number @r number @g number
  @b )
  #Solid Line
  colour ( RGBi ( @r @g @b ) )
  linestyle ( 0 0 0 ( ) )
  path ( ( -12 -7 -12 10 12 10 12 -7 close ) )
  edge ( )
endprocedure

```

```

#
# A left to right slash
#
procedure _lslash marker ( number @r number @g number @b
)
  #Solid Line
  colour ( RGBi ( @r @g @b ) )
  linewidth ( 0.3 )
  linestyle ( 0 0 0 ( ) )
  path ( ( -10 -5 10 5 ) )
  edge ( )
endprocedure

```

```

#
# A right to left slash
#
procedure _rslash marker ( number @r number @g number @b
)
  #Solid Line
  colour ( RGBi ( @r @g @b ) )
  linewidth ( 0.3 )
  linestyle ( 0 0 0 ( ) )
  path ( ( -10 5 10 -5 ) )
  edge ( )
endprocedure

```

```

#
# Infantry: is a centered cross (2 slashes)
#
procedure infantry marker ( number @r number @g number
  @b )

```

```

#
#
#
#

```



FIGURE 5.7

```

  path ( ( 0 0 ) )
  #Solid Line1
  _lslash ( @r @g @b )
  path ( ( 0 0 ) )
  #Solid Line2
  _rslash ( @r @g @b )
endprocedure
#
# Armor: is a centered ellipse
#
procedure armor marker ( number @r number @g number @b )
#
#
#

```



FIGURE 5.8

UNCLASSIFIED

37

```

colour ( RGBi ( @r @g @b ) )
linewidth ( 0.3 )
linestyle ( 0 0 0 ( ) )
#Solid Line1
path ( ( -5.5 3 5.5 3 ) )
edge ( )
#Curve line1
path ( ( 5.5 3 arccw 5 0 5.5 -3 ) )
edge ( )
#Solid Line2
path ( ( -5.5 -3 5.5 -3 ) )
edge ( )
#Curve line2
path ( ( -5.5 -3 arccw -5 0 -5.5 3 ) )
edge ( )
endprocedure

```

One of the ACTIF requirements is the ability to attach textual information to a symbol. 'Graphics' supports Adobe Type 1 fonts through a font server called «xfs» (see Annex A – The Inside Stuff).

Note: We did not have the opportunity to test the following graphics text functions because the «xfs» was not running properly under GenaMap 7.1 b3. The customer support told us that it was already corrected and well operational under the final release of GenaMap 7.1.

Any Type 1 fonts can be added to a symbol using the same «primitive» technique. Some graphics functions implemented in «primitive» let us manage strings like: 'textfont', 'textsize', 'textalign', 'textrotation', 'textdraw', etc.

```

#-----
#
# File:                military.symbols
# Description:         This file contains a test base
#                      for military symbols
#                      used by the Genasys family of
#                      product.
# Creation Date:       This file was created by FAUDET on
#                      April 97
# Last Modification:   none.
#-----

```

```

#-----
#
# Procedure linked to field «B» symbol of the Military
# Symbols APP-6
#
#-----
#
#
#
#

```



81

FIGURE 5.9

```

# Size indicator and Rating example
#
procedure size marker ( )
    path ( ( 0 0 ) )
    textsize ( 3 3 )
    textalign ( 1 )
    textfont ( Times-Roman )
    textdraw ( I I )
    textalign ( 6 )
    textdraw ( 81 )
endprocedure

```

Before any symbol display, we need to create a library containing the military symbol representation. Creating all of the possible symbols in a static file represents a huge amount of work. We decided to implement our library in a more dynamic way. Thus, each symbol is built from our «primitives» pool and saved in a common ASCII file constituting a «dynamic library». This library could be preserved along with session parameters. Modifications of the «primitives» file will allow future symbol evolution without rebuilding any related library. Any modification of the «dynamic library» during run time will require a refresh of the «library handle» illustrated in the following script:

```

#-----
#
# File: refresh
# Description: Script to refresh connection to the
#              military symbols library.
#
# Creation Date: This file was created by FAUDET on
#               April 97
# Last Modification: none.
#

```

UNCLASSIFIED

39

```

# Input:          Handle $1
#-----

#
# Close an open symbol library
#
address 45000 grcloselib $1 -library/
        export/home/aufl/extension/graphics/
        military.symbols

#
# Open an existing symbol library
#
address 45000 gropenlib $1 -library/
        export/home/aufl/extension/graphics/
        military.symbols

```

There are two available functions when displaying a symbol vector file with GenaMap 7.1. The first one requires you to draw each specific symbol one at a time to their location using 'grdraw' function. Alternatively, you could use 'gmplot handle vector_map symbol_id' to display identical features, like a river map, with the same symbol representation (blue line). Those approaches do not exactly meet the ACTIF needs. The GIS engine should take care of the marker representation allowing fast display and refresh by internally assigning symbols to their physical locations. Currently, GenaMap 7.1 does not have a way to assign features to new symbology definitions. One way to bypass this limitation is to give our symbol names the same names that gmap.symbols (gmap.BT) does (ASSIGNEDMARK1, ASSIGNEDMARK2, etc.). We could also create new bundle tables (refer to previous sub-section GenaMap 6.2) with the index number corresponding to our «procedure» name. Finally, using the GenaMap command 'assign', we have been able to specify a symbol representation from our own library for each point, line and area feature our map contains.

Once built and assigned, dynamic symbols are available for display on a situation map. In Section 8 we will discuss in detail the initial display

process. Nevertheless, for the GenaMap 7.1 Graphics service evaluation, we illustrated it with a script:

```
#-----
#
# File:          tactical
# Description:   Script to initialize tactical
#               symbology.
#
# Creation Date: This file was created by FAUDET on
#               April 97
# Last Modification: none.
#
# Input:         Handle value $1
#-----
#
# Add a viewport to our existing window
#
read -l «$(getv wind)» xmin xmax ymin ymax rest
read -l «$(address 45000 grgetpagesize $1)» width height
address 45000 graddvp $1 -worlds $xmin $ymin $xmax $ymax
      -into 0 0 $width $height
#
# Draw assigned symbols
#
gmplot $1 northwest ASSIGNED
```

Finally, «Graphics» service offers many functions to control the appearance or behaviour of the symbols. A conditional graphics primitive gives to the symbol representation some decision-making ability using what is called the «intelligent symbology». This, combined with expressions and parameters, allows a symbol to be displayed differently depending upon circumstances. It means that support for 'de-cluttering' to aid visibility at certain map scales can be easily implemented. Moreover, the function 'grsymbolscale', combined with a «primitive» definition 'size', specify the scaling factor for a symbol as illustrated in Figures 5.10 and 5.11. All symbol sizes will be multiplied by this scale factor thus implementing the concept of «scale of true display».

5.4 Recommended Approach

At the time we wrote this document, it was clear that Genasys 7.1 (Graphics and Glyph modules) will fully support all symbology needs for ACTIF development. Thus, the basics established through our scripting approach will remain and could be implemented in an object-oriented architecture.

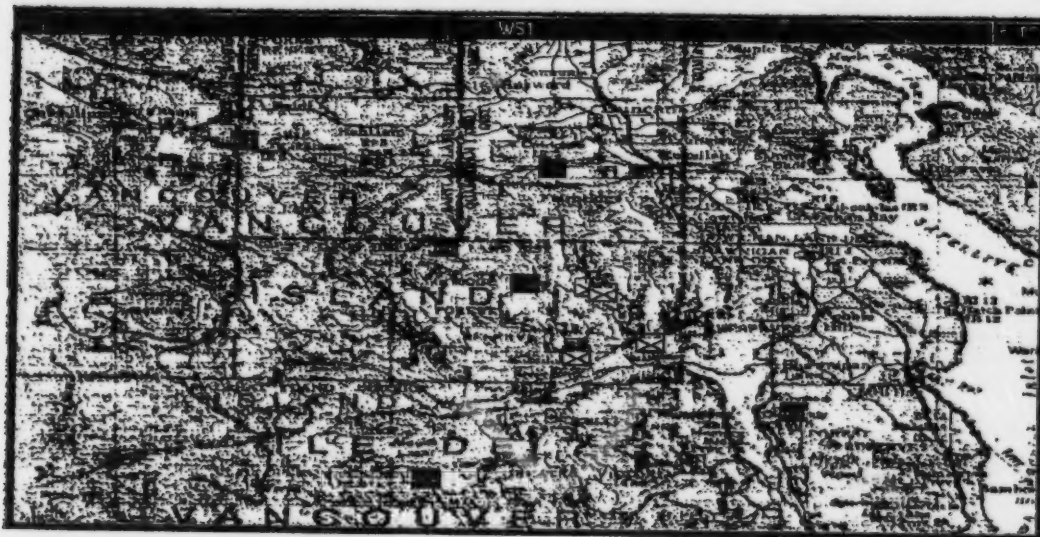


FIGURE 5.10- Initial display of tactical data

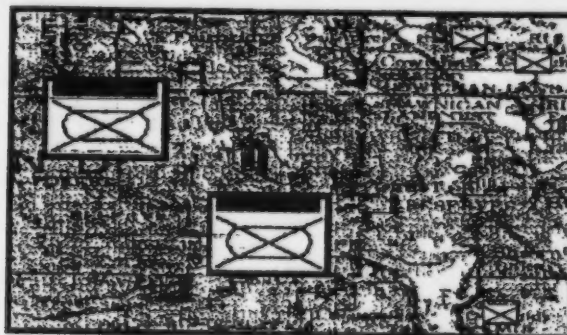


FIGURE 5.11- Scalability of symbols

5.5 Implementation Rule

There are no rules to mention for this function.

6.0 SIMULATE THE CLASS ENCAPSULATION OF RDBMS

6.1 Goal

It is essential for a good command and control information system (C2IS) to possess the capacity to access data through a Relational Data Base Management System (RDBMS). Therefore in this context, it is important to explore the capability of a GIS to access a database by clicking on a military unit symbol on a working situation map. Some issues like response time and robustness were considered.

6.2 Result Summary

This method of accessing the Oracle database has been successfully implemented. Even if Genasys provides a bridge to access RDBMS (Relational Data Base Management System), it is an approach that has not been considered. For architecture compliance, the class encapsulation has to be carried out directly in the «generic» part of the C++ code. We experienced problems regarding the reliability of our connection with SQL*net due to the short testing time used for our development process. We do not foresee any particular problems regarding the encapsulation of RDBMS.

6.3 Approach Used

Despite the fact that we can use Genasys to connect to our database, we decided to re-use the knowledge of the ACTIF architecture development to implement a «three-tier» approach. The graphical user interface tier is responsible for the display. The application tier is the main core of the ACTIF system. The database tier may be either a collection of raw data (a data store) or a database management system. Thus, we did detach all database access from the GIS environment. A simple GUI is used to manipulate the symbols (Figure 6.1): uiObject.cpp class.

The Object class methods and attributes are handled by the Object.pc and access to Oracle: selection, insertion and update are encapsulated in dbObject.pc. We use PRO-C to pre-compile Object.pc and dbObject.pc into a super class called Object.cpp.

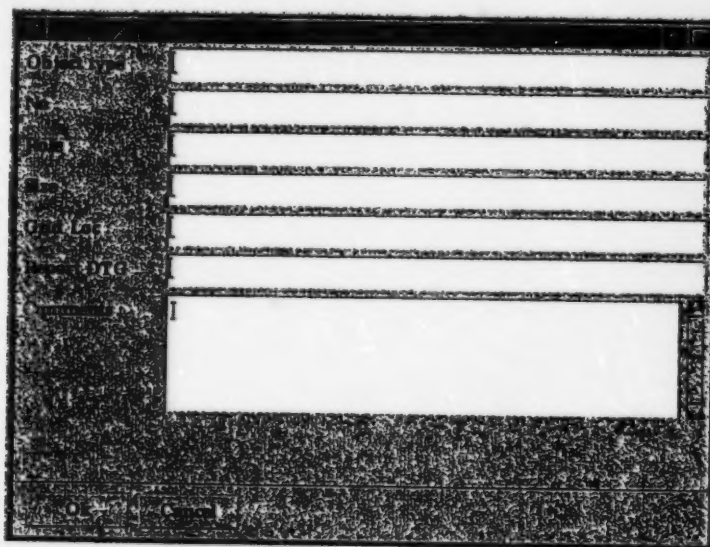


FIGURE 6.1- Symbology Dialog

TABLE 6.1: FIELD DESCRIPTION

Field	Description
Object type	25 characters describing an object (symbol).
No	25 characters identifying an object.
Role	25 characters giving an object role.
Size	25 characters describing the size of an object.
Grid Loc	This specific class will handle the grid location of an object (future development). Hard coded read-only for the prototype version.
Report DTG	25 characters indicating the date time group.
Comments	255 characters.

TABLE 6.2: BUTTON DESCRIPTION

Button	Description
[OK]	Submit the data to the database.
[Cancel]	Undo the modification.

6.4 Recommended Approach

The results of our approach lead us to think that duplication of data will not be necessary. The performance was not an issue and we expected with this approach that there will not be any Genamap database connection within the application. The «three-tier» approach used in the prototype is recommended for ACTIF.

6.5 Implementation Rules

The reader is advised to refer to "Coding Classes" of section 6 from Internal Design Standard [18].

7.0 INITIAL DISPLAY OF TACTICAL DATA

7.1 Goal

It is essential for a good command and control information system (C2IS) to possess the capacity to start-up quickly when the application is called. Therefore in this context, it is important to explore the capability of a GIS to bring up a «Senior analyst» role and to rapidly process all the tactical database information from Oracle and display the resulting symbols (units) on the situation map. Please refer to the Figures 5.10 and 7.1.

7.2 Result Summary

The dynamic creation of a vector file has been successfully implemented with the use of «Data Encapsulation» API commands. Data are stored and managed in Oracle but are generated (in vector) only at the beginning of a given session. To achieve the implementation of such a solution, the software must provide good performance for the generation of tactical information. GenaMap provided the kind of functionality and performance that was required.

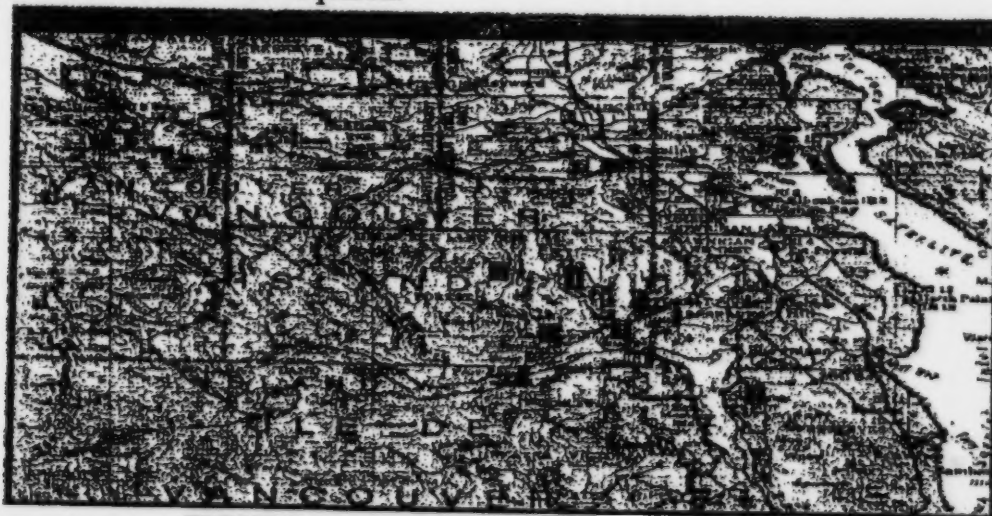


FIGURE 7.1- Situation map WS1 and symbols

7.3 Approach Used

We decided to work directly in the GenaMap vector file type 4 using what is called the «Data Encapsulation» approach. The idea is to use the knowledge gained from section 6 to dynamically create a vector file reflecting the contents of an Oracle database. The creation of a vector file takes these steps: create a map header object, create a vector map header object and then create an empty vector file. Once the map vector file is created and opened, we insert a node for each unit (record) found in Oracle where we attach a point feature. The point feature uses a «TAG» to identify itself and a symbol number 260 (■) is associated from the bundle table FONTS. The «CreateTactical» method is responsible for the initial display of tactical data:

```
WorkingSitMapView::CreateTactical()
{
    GSDB      *sdb;
    GVM_HEAD  *vecthd;
    GMAP_HEAD *mapinfo;
    GVM_FP     *vectfp;
    GVM_NODE   *node;
    GVM_POINT  *point;
    int        node_id;
    GMBR_D      mbr;
    GPROJECTION projection;

    /*
     * Delete previous type 4
     */
    system («rm -r ZF04/cotel»);

    /*
     * Connect to the Database
     */

    if( (sdb= GSDBConnect («/export/home/aufl/
    actifPrototype/CommonFeatures/Application/exe», «»,
    «»)) ==NULL)
    {
        fprintf(stderr, «Error connecting database\n»);
        return(0);
    }

    /*
```

UNCLASSIFIED

47

```

* Set the mbr values
*/
    mbr.xmin = -124.0015869;
    mbr.ymin = 47.9999695;
    mbr.xmax = -120.0010986;
    mbr.ymax = 50.0000992;
    mbr.zmin = mbr.zmax = 0.0;

/*
* Set the projection value.
*/
    projection.code = GCTP_SYS_GEOGRAPHIC;
    projection.zone = 0;
    projection.units = GCTP_DEGREES;
    projection.ellipse = GCTP_ELL_CLARK1866;

/*
* Create a map header object.
*/
mapinfo = GInstantiateMapHeader(
    «cotel»,
    GVECTOR_2D,
    «Vector map representing the initial
    display»,
    250000,
    GTHIRTY_TWO_BIT,
    &mbr,
    &projection
);

/*
* Create a vector map header object.
*/
if( (vecthd = GVMInstHeader( mapinfo))==(GVM_HEAD*)0)
{
    fprintf(stderr, «Error (%d): %s\n», GErrorNum(),
        GErrorMess() );
    return(0);
}

GFreeMapHeader(mapinfo);

/*
* Create a new vector map
*/
if( !GVMMCreate(sdb, «cotel», vecthd) )
{
    fprintf(stderr, «Error (%d): %s\n», GErrorNum(),
        GErrorMess() );
    return(0);
}

```

SANS CLASSIFICATION

```

/*
 * Open specified vector map partition.
 */
if( (vectfp =
GVMOpen(sdb,«cotel»,GREAD_WRITE))==NULL)
{
    fprintf(stderr, «Error opening the vector map\n»);
    exit(1);
}

if( !iOpenCursor() )
{
    while( iFetch() )
    {
        /*
         * Instantiate GVM_NODE object.
         */
        if( (node = GVMAllocNode())==NULL)
        {
            fprintf(stderr, «Error allocating
            node:\n %d%s\n»,
            GErrorNum(), GErrorMess());
            return(0);
        }

        /*
         * Get node header info.
         */
        node->x      = strDbNode.fPosX;
        node->y      = strDbNode.fPosY;

        /*
         * Insert GVM_NODE object into partition.
         */
        if( (node_id = GVMInsertNode(vectfp,
        node)) < 1)
        {
            fprintf(stderr, «Error inserting
            node: \n%d%s\n»,
            GErrorNum, GErrorMess());
            return(0);
        }

        /*
         * Free allocated node.
         */
        GVMFreeNode(node);

        /*
         * Instantiate GVM_POINT object.
         */
        if( (point = GVMAllocPoint())==NULL)

```

UNCLASSIFIED

49

```

{
    fprintf(stderr, «Error allocating
point:\n %d%s\n»,
GErrorNum(), GErrorMess());
    return(0);
}

/*
 * Get point header info.
 */
point->node_id = node_id;
point->symb_id = 260;
sprintf(point->tag.string, «%d»,
strDbNode.iPosId);
point->tag.x      = strDbNode.fPosX;
point->tag.y      = strDbNode.fPosY;
point->tag.disp_x = strDbNode.fPosX;
point->tag.disp_y = strDbNode.fPosY;

/*
 * Insert GVM_POINT object into
partition.
 */
if( (GVMInsertPoint(vectfp, point)) < 1)
{
    fprintf(stderr, «Error inserting
point: \n%d%s\n»,
GErrorNum(), GErrorMess());
    return(0);
}

/*
 * Free allocated point.
 */
GVMFreePoint(point);
}

}

iCloseCursor();

/*
 * Close map.
 */
GVMClose(vectfp, TRUE);

/*
 * Disconnect from database.
 */
GSDBDisconnect(sdb);
};

```

SANS CLASSIFICATION

Statistics

Demo 1: The total initial display time was 22 seconds on a Sparc station 5. We used two Oracle tables: OBJECT and POSITION containing 300 records each. The Oracle connection to SQL*net was establish at the beginning of the «IntManager» prototype. The «CreateTactical» method adds 2 to 3 seconds to the total initial display time.

Demo 2: The total initial display time to «cache» our three map scales was eight minutes on a Sparc station 5 (three minutes on a Sparc Ultra 140).

7.4 Recommended Approach

It is clear that, from the point of view of symbology, our prototyping approach was simple. The ACTIF development will add some processes to create object oriented symbols (the reader may revisit Section 5). The point feature will be replaced by the appropriate symbol on the map. Nevertheless, the creation of the Genasys type 4 file will remain our preferred approach.

7.5 Implementation Rules (Annex B)

The reader will find the implementation rules and the necessary scripted routines at Annex B to perform the function: Creating type 4 maps.

8.0 TOOLBOX – SELECT AND QUERY FUNCTIONALITY

8.1 Goal

It is essential for a good command and control information system (C2IS) using a GIS to provide easy access to symbol information displayed on electronic maps. Therefore in this context, it is important to explore how rapidly a user can access symbol information at a specific location, display the information in an independent window and update the information field if required.

8.2 Result Summary

The select and query portion of functionality identified in the previous section (Section 7 – Initial display of tactical data) has been successfully implemented. Because different alternatives for initial display exist, these select and query functionality could have become issues. The simple solution retained in Section 7 made the implementation of Select and Query functions easier. With the use of these tools, the user can select and query any feature in the generated file.

8.3 Approach Used

The implementation of the querying functionality was done at the «Command Encapsulation» level. We used the knowledge gained in Section 6 to retrieve the database information. The GenaMap function 'query tag' was implemented in the toolbox button «? Obj». This function allows the system to obtain the internal tag number of the nearest feature within a search criterion. Each time the toolbox button «? Obj» is activated, the user enters a query (input) mode. This mode allows the user to locate any number of objects on the situation map by clicking the left mouse button. Each click displays the search criterion around the pointer location. When finished, the

user must click the right mouse button to quit the input mode and retrieve the corresponding information. Using the tag number, a selection is made in the database and the result (if any) is output via the common graphical user interface.

```
void WorkingSitMapView::vQueryObject()
{
    MyObject *anObject;
    int unitId=0;
    BOOLEAN status;
    MMESSAGE *message;
    char msgContainer[2048];

    execute_genamap_command(«query tag»);

    for (int i=0; (status=GFetchMessage( &message )) &&
        message != NULL;i++)
    {
        if( GSprintfMsg(msgContainer, message) > 0)
        {
            switch(i)
            {
                case 1: sscanf(msgContainer, «%d»,
                    &unitId);break;
                default: break;
            }
        }
        else
        {
            fprintf(stderr,«Could access
            message.\n»);
            return;
        }
    }
    if( !status ) {
        printf («Error from GFetchMess():\n%d  %s\n»,
            GErrorNum(), GErrorMess() );
        return;
    }
    if( unitId )
    {
        anObject = new MyObject(unitId);
        anObject->vBuildWindw();
        anObject->gui->show();
    }
};
```

8.4 Recommended Approach

The functional aspect of the future system will clearly establish the Select and Query functions. At the time we wrote this document, our preferred approach was to use the tag number as the key to all our processing between Oracle, the GIS and the ACTIF system. Nevertheless, the final database and functional architecture could lead us to modify our approach.

8.5 Implementation Rules (Annex B)

There are no rules to mention for this functionality.

9.0 TOOLBOX – PLACE UNIT SYMBOL FUNCTIONALITY

9.1 Goal

The work of a "Senior Analyst" implies introducing hypotheses into the ACTIF system. Those hypotheses involve creating and inserting tactical symbols onto a working situation map. Clicking on the map symbol must access information about the symbol in an associated database. This section uses a different approach from section 6 to implement the 'insert' functionality.

9.2 Result Summary

The insert function must manage on one hand the insertion of the vector type structure onto the map and on the other hand insert a feature's characteristics into the RDBMS. For example, the feature coordinates must be calculated from the feature position and then inserted into the RDBMS. The development team was able to prototype all the required functionality to insert and update features.

9.3 Approach Used

Creating units was illustrated in the prototype by the «Insert» button. This button implements the insertion of a unit symbol type. This process involves «Command Encapsulation» as explained in Table 9.1. The vector map is then redrawn. The new inserted object is displayed using symbology number 260 from the bundle table FONTS.

```
void WorkingSitMapView::vInsertObject()
{
    MyObject *anObject;

    execute_genamap_command («edmap cotel; gloc;
    edbn;»);

    anObject = new MyObject(this);
    anObject->vBuildWindw();
}
```

```

anObject->gui->show();
};

```

TABLE 9.1: COMMAND ENCAPSULATION PROCEDURE

ACTION	DESCRIPTION
1- The user clicks on the «Insert» button.	The WorkingSitMapView::vInsertObject method is called.
2- The user positions the symbol on the working situation map.	The functions 'edmap cotel; gloc; edbn;' are executed at the «Command Encapsulation» level.
3- Unit symbol information is input using the common graphical interface. Refer to Figure 6.1.	The mandatory fields are Object type, No, Role, Size, Report DTG and comments.
«Cancel» button is clicked.	The function 'edbundo' is executed.
«OK» button is clicked.	The function 'edbttag sequence; edbsave' is executed and the database is committed.

9.4 Recommended Approach

The functional aspect of the future system will clearly establish the 'insert unit' functionality. At the time we wrote this document, we decided to use another approach from the initial display of tactical data (Section 7). Thus we used the «Command Encapsulation» to update our GenaMap type 4 file. Nevertheless, the final database and functional architecture may lead us to modify our approach and use what is called «Data Encapsulation» (refer to Section 6).

9.5 Implementation Rules

There are no rules to mention for this functionality.

10.0 SITUATION MAP REFRESHES

We did identify two issues when working on a situation map. The first one was the initial display of the map (raster, vector and symbol data). It was addressed in Section 2 and 7. The second one is to refresh part or all the information contained in a "drawable" window of a situation map.

Note: We did not have time to complete this section within the time frame of the prototype development but we do not foresee any particular problem.

11.0 COORDINATE MANAGEMENT (MAP PROJECTION, ETC.)

The situation map will require transformation of user-entered coordinate values from one map projection to another. GenaMap offers thirty-five (35) coordinate systems (projections) and twenty-three (23) ellipsoids. Thus, we believe that it will be possible to encapsulate the Military Grid Reference System (MGRS) within the ACTIF system.

Note: We did not have time to complete this section within the time frame of the prototype development but we do not foresee any particular problem.

12.0 AREA SYMBOLS DISPLAY AND PLACEMENT

In Section 9, we talked about inserting symbols on the situation map. Part of the work of a «Senior Analyst» is also inserting symbols related to terrain features. Within GenaMap, some basic commands were grouped to form a script called 'digitize'. This script gives a user the ability to insert lines and area features on a vector map. Thus, we believe that it will be possible to encapsulate the basic commands within the ACTIF system to meet our requirement.

Note: We did not have time to complete this section within the time frame of the prototype development but we do not foresee any particular problem.

13.0 CONCLUSION

13.1 Context

The mission of the intelligence and surveillance combat function is to provide friendly forces with timely, accurate intelligence about hostile dispositions, capabilities and intentions, geographic conditions, targets and meteorology, while also denying friendly force information to an opponent. The provision of timely and accurate intelligence is an essential component in the Commander's decision-making process and the synchronization of battlefield combat power. It is a key element in the planning and conduct of all military operations. Thus, the rapid passage of information and intelligence to all levels of command is a vital and continuous activity.

Nowadays, effective computing support is needed to cope with the large flow of information generated by modern surveillance systems and troops in contact. Furthermore the uncertain character of the information such as inaccuracy, incompleteness and redundancy greatly complicates the task of timely intelligence analysis and strengthens the need for information technology. Also the priority of effort for the Canadian Forces has shifted from Central Europe, where a well-defined enemy and known mission fostered fixed organizations and standard operating procedures, to a less certain situation where neither the mission nor the enemy are known in advance. An Army intelligence system must be capable to quickly adapt to these new evolving situations and carry out its primary tasks in a flexible manner with a tailored intelligence capability appropriate to each mission element.

Any automated Command and Control Information System (C2IS), in order to be successful, must rely on an efficient geographical information

system (GIS). Therefore, the purpose of this Technical Memorandum (TM) is to present the results of prototyping activities aimed at exploring one technological alternative with the Genasys family of products.

13.2 Results

This study has demonstrated that the Genamap family of products would meet ACTIF needs. The following summarizes the strengths and weaknesses identified.

13.2.1 Strengths

RELIABILITY

We do not foresee any problem regarding the reliability of the product. To achieve the level of robustness that ACTIF needs, the GenaMap family of products will have what we are looking for. Simple programming schemes will have to be designed and followed by all developers.

DISPLAY SPEED

During the current project, two prototypes were developed using ADRG raster files. They demonstrated that the display refresh could be accomplished under one second using the modules provided with Genamap with of course some optimization.

SYMBOLLOGY

Genamap symbology services will fully support all symbology needs for ACTIF development, including the need to be fully compliant with the military specifications.

SUPPORT

This prototyping activity showed that the Genamap family of products are sufficiently documented and their customer support is efficient.

13.2.2 Weaknesses

INTEGRATION

The Genamap family of products are designed to work together via a «Request Broker» architecture. This kind of open architecture, when not required, brings integration complexity and maintenance difficulties.

COST

Finally, the customer support cost and the acquisition cost are relatively high.

13.3 Final Note

The Defence Research Establishment, Valcartier (DREV) has designed and produced automated tools to support the development of a modern tactical intelligence processing and analysis capability for the Canadian Land Force. Under the DREV "Advanced Concepts for Tactical Information Fusion" (ACTIF) project many studies were conducted to assist in the design of a future automated tactical intelligence system [5-10]. A new major R&D project D6479 LIEWA will contribute to define, validate and integrate concepts and requirements into Intelligence and EW technology demonstrators supporting the LF ISTAR architecture. This development is required due to an acknowledged Canadian Forces (CF) need for an intelligence and information technology investigator and evaluator to help reduce implementation risks of the capital acquisition L2634 ATISS project. It is hoped that the results of this study constitute a valid contribution that will enable the Intelligence Branch organization to produce a detailed design, to deliver and to implement a Tactical Intelligence System capable of embracing the Information Age.

REFERENCES

- 1- Gauvin, M., Lapointe, S. and Lizotte, M., "Tactical Information Fusion Prototype: Evaluation Through an Abbreviated Command Post Exercise", DREV R-9503, September 1995, UNCLASSIFIED
- 2- Gauvin, M. and Thibault, G., "Tactical Information Fusion Prototype: Participation in the ABCA Northern Lights Command Post Exercise", DREV R-9531, January 1996, UNCLASSIFIED
- 3- Thibault, G. and Beaumont, G., "Katimavik Project Phase I: Proposed TIF Prototype Modifications", DREV TM-9511, April 1996, UNCLASSIFIED
- 4- Thibault, G. and Beaumont, G., "Exercise Rite Complex 95, TIF Prototype Users' Comments", DREV TM-9511, August 1996, UNCLASSIFIED
- 5- "Development of a Strategic Plan for a Tactical Intelligence System", Final Report, DMR Consulting Group Inc., Contract W7701-6-2782, November 1997, UNCLASSIFIED
- 6- "Technological Infrastructure Architecture for a Tactical Intelligence System", Final Report, DMR Consulting Group Inc., Contract W7701-6-2782, November 1997, UNCLASSIFIED
- 7- "ACTIF Architecture, Database Design Description", Volume 1 and 2, DMR Consulting Group Inc., Contract W7701-6-4114, March 1997, UNCLASSIFIED
- 8- "ACTIF Architecture, System-Sub-System Design Description", DMR Consulting Group Inc., Contract W7701-6-2782, November 1997, UNCLASSIFIED
- 9- "ACTIF Architecture, Functional and Internal Standards", DMR Consulting Group Inc., Contract W7701-6-2782, November 1997, UNCLASSIFIED
- 10- "Concepts, Definitions and Requirements for a Future Tactical Intelligence System", DREV TM 9833, January 1999, CONFIDENTIAL

UNCLASSIFIED

61

- 11- "GIS Study - Selection of technical Alternatives", ACTIF System Architecture Collection, DMR Consulting Group Inc., Contract W7701-5-1076, March 1996, UNCLASSIFIED
- 12- "GIS Study Part II - Selection of technical Alternatives", ACTIF System Architecture Collection, DMR Consulting Group Inc., Contract W7701-5-4114, December 1996, UNCLASSIFIED
- 13- Gibb, A., Tremblay, J., Turcotte, L., and Audet, F., "Selection of a Commercial GIS for Integration into a Command and Control Application - Lessons Learned from the ACTIF Intelligence Automation Project", DREV Technical Memorandum TM-9810, May 1998, UNCLASSIFIED
- 14- Graphics Reference Manual, Draft Version 1.0, Genasys II, December 1996, UNCLASSIFIED
- 15- B-GL-303-002/FP-Z01 Military Symbols, Volume 2, Supplement 1, Issued on Authority of the Chief of the Defense Staff, Canada, January 1989, UNCLASSIFIED
- 16- The Displayer Reference Guide, Version 1.2, Genasys II, October 1995, UNCLASSIFIED
- 17- GenaVive Registration Utility User Manual, Version 2.1, Genasys II, October 1995, UNCLASSIFIED
- 18- Internal Design Standard, Standards for Software Design, prepared by DMR Consulting Group, Sainte-Foy (Québec) under contract W7701-5-4114, October 1996, UNCLASSIFIED
- 19- Cockcroft, Adrian, "Sun Performance and Tuning", SunSoft Press, USA, 1995, UNCLASSIFIED

SANS CLASSIFICATION

ANNEX A – THE INTERNAL STRUCTURE

A.1 Directory

The prototype main program is located in the **actifPrototype / CommonFeatures / Application** and the GIS object classes are located in the **actifPrototype/SituationMap/Gis** directory.

A.2 Library

The prototype library is located in:

actifPrototype/SituationMap/Gis /lib/Gis.a

The symbol library is located in:

workarea /graphics/military.symbols

TABLE A.1: FILE DIRECTORY

Directory actifPrototype/SituationMap/Gis/src/	Description
SpatialView.cpp	Intended to reflect the C32_SpatialView of O_F32_VIEWS
MapView.cpp	Intended to reflect the C32_MapView of O_F32_VIEWS
SitMapView.cpp	Intended to reflect the C32_SitMapView of O_F32_VIEWS
UiWorkingSitMapView.cpp	This class represent the toolbox GUI designed for Demo 1
WorkingSitMapView.pc	This Pro*C file encapsulates the class C32_WorkingSitMapView of O_F32_VIEWS. This is actually where all the Demo 1 functionality was created.
DbWorkingSitMapView.pc	This Pro*C file encapsulates the Oracle access for the initial display of the tactical information.

UNCLASSIFIED

A-2

WorkingSitMapView.cpp	This class results from the pre-compiled WorkingSitMapView.pc and DbWorkingSitMapView.pc
uiObject.cpp	This class is used to display standard information of all symbols on the WorkingSitMapView
Object.pc	This Pro*C file encapsulates the class MyObject designed to Insert, Update and Query symbols on the WorkingSitMapView.
DbObject.pc	This Pro*C file encapsulates the Oracle access for Insert, Update and Select.
Object.cpp	This class results from the pre-compiled Object.pc and DbObject.pc
uiOverlayList.cpp	This class was designed to control the visibility of the layers(GUI). Not completed. Not yet implemented in any Demo.
OverlayList.cpp	This class was designed to control the visibility of the layers(kernel). Not completed. Not yet implemented in any Demo.
ShownOverlay.cpp	This class was designed to control the visibility of the layers(layer object). Not completed. Not yet implemented in any Demo.

A.3 Makefile

To create the Gis.a library you need to create an appropriate Makefile in the `actifPrototype/SituationMap/Gis/src/` directory:

```
include $(MAKE_DIR)/MakeActifVars

LIBNAME= Gis

PROCESSES=

CLASSES= SpatialView ShownOverlay uiOverlayList
          OverlayList MapView SitMapView
```

UNCLASSIFIED

A-3

```
uiWorkingSitMapView WorkingSitMapView uiObject Object
```

```
SPEC_INCS= $(APPLICATION_INCS) $(INTEGRATION_INCS)
           $(UTILITIES_INCS) $(GUI_INCS)
           $(API_INCS)
```

```
include $(MAKE_DIR)/MakeActif
```

To compile a Spatial API or a Dispatcher API program like the prototype, be sure to have the following lines in your Makefile:

actifPrototype/actifUtils/MakeActif:

```
$(EXE_DIR)/%: $(BIN_DIR)/%.o $(SYS_LIBS) $(EXE_DIR) \
              $(INTEVENTS_DIR)/lib/IntEvents.a \
              $(ORGANIZATION_DIR)/lib/Organization.a \
              $(INTEGRATION_DIR)/lib/Integration.a \
              $(MESSAGES_DIR)/lib/Messages.a \
              $(GUI_DIR)/lib/GUI.a \
              $(UTILITIES_DIR)/lib/Utilities.a \
              $(FILTER_DIR)/lib/Filters.a \
              $(GIS_DIR)/lib/Gis.a \
              $(LIB_DIR)/intManager.a
...
@ if $(PURIFY) $(CPP) $(CFLAGS) $(CPPFLAGS) -o $@
    $(LIBPATH) $(BIN_DIR)/
$*.o $(API)/map/lib/gmapgb1.o $(API)/map/lib/sysblk.o
    $(ACTIF_LIBS) $(ORACLE_LIB
S) $(MOTIF_LIBS) -lxview -lglx -lX11 -lm ; \
```

actifPrototype/actifUtils/MakeActifVars:

```
API_INCS = $(API)/map/include $(API)/include
           $(GENA)/dispapi_6.2/include
API_LIBS = \
```

```
$(API)/map/lib/libapi.a \    $(API)/lib/libscript.a \    $(API)/map/lib/libmap3.a
                             \
$(API)/map/lib/libmap2.a    $(API)/map/lib/libmap1.a \    $(API)/map/lib/libmap0.a
                             \
$(API)/map/lib/libnetwork    $(API)/map/lib/libselect.a \    $(API)/map/lib/libtext.a
.a \                             \
$(API)/map/lib/libexdb.a      $(API)/map/lib/libhrdcpy.a \    $(API)/map/lib/libmap4.a
                             \
```

UNCLASSIFIED

A-4

```

$(API)/map/lib/libedmap.a $(API)/map/lib/libfont.a \ $(API)/map/lib/libxcell.
\ a \
$(API)/map/lib/libcell.a $(API)/map/lib/libuserio.a \ $(API)/map/lib/libreform
\ .a \
$(API)/map/lib/libaid.a \ $(API)/map/lib/libmap3.a \ $(API)/map/lib/libmap2.a
\ \
$(API)/map/lib/libvectio. $(API)/map/lib/libmap1.a \ $(API)/map/lib/libfilemg
a \ t.a \
$(API)/map/lib/libmap3.a $(API)/map/lib/libfilemgmt.a \ $(API)/map/lib/libmap0.a
\ \
$(API)/map/lib/lib symb.a $(API)/map/lib/libcnvrt.a \ $(API)/map/lib/libproj.a
\ \
$(API)/map/lib/librtree.a $(API)/map/lib/libtif.a \ $(API)/map/lib/libmap0.a
\ \
$(API)/map/lib/libcnvrt.a $(API)/map/lib/libisam.a \ $(API)/map/lib/libbtree.
\ a \
$(API)/map/lib/libtif2.a $(API)/map/lib/libgraphics.a \ $(API)/map/lib/libtext.a
\ \
$(API)/map/lib/libuserio. $(API)/map/lib/libdevio.a \ $(API)/map/lib/libpipe.a
a \ \
$(API)/map/lib/libutils.a $(API)/map/lib/libbyte.a \ $(API)/map/lib/libsys.a
\ \
$(API)/map/lib/libmessage $(API)/map/lib/libwindows.a \ $(API)/map/lib/libtools.
.a \ a \
$(API)/map/lib/libstubs.a $(API)/lib/libprotect.a \ $(API)/lib/libdbclnt.a \
\ \
$(API)/lib/libclient.a \ $(API)/lib/librpc_tcp.a \ $(API)/lib/libserver.a \
$(API)/lib/libcmd.a \ $(GENA)/dispapi_6.2/lib/libdispa $(API)/lib/libdisp.a \
pi.a \
$(API)/lib/libg2cogo.a \ $(API)/lib/libxutil.a \ $(API)/lib/libg2io.a \
$(API)/lib/libg2msg.a \ $(API)/lib/libg2misc.a \ $(API)/lib/libg2math.a

```

```

F77LIBS = -lm77 -lf77 -lsunmath
GR_LINK_LIBS = \

```

```

$(API)/lib/libfgraph.a \ $(API)/lib/libgraph0.a \ $(API)/lib/libx11.a \
$(API)/lib/libhpgl.a \ $(API)/lib/libgraph1.a \ $(API)/lib/libxutil.a \
$(API)/lib/libg2misc.a \ $(API)/lib/libg2math.a

```

```

GIS_LIBS= $(GIS_DIR)/lib/Gis.a

```

SANS CLASSIFICATION

UNCLASSIFIED

A-5

```
GIS_INCS= $(GIS_DIR)/include $(COMMON_INCS)
```

...

```
ACTIF_LIBS= $(APPLICATION_DIR)/lib/intManager.a  
$(COMMON_LIBS) $(MESSAGES_LIBS)  
$(ORGANIZATION_LIBS) $(INTEVENTS_LIBS)  
$(INTEGRATION_LIBS) $(FILTER_LIBS)  
$(UTILITIES_LIBS) $(GIS_LIBS) $(API_LIBS)  
$(F77LIBS) $(GUI_LIBS) $(GR_LINK_LIBS)
```

...

```
PROCLFLAGS= ireclen=255 oreclen=255 code=cpp  
select_error=no $(SQLCHECK) $(PROUSER) dbms=v7  
SYS_INCLUDE=\ (/opt/SUNWsprow/SC4.0.1/include/CC,/usr  
/include\) include=$(APPLICATION_DIR)/include  
include=$(GUI_DIR)/include  
include=$(MESSAGES_DIR)/include  
include=$(INTEVENTS_DIR)/include  
include=$(FILTER_DIR)/include  
include=$(INTEGRATION_DIR)/include  
include=$(ORGANIZATION_DIR)/include  
include=$(UTILITIES_DIR)/include  
include=$(MOTIFHOME)/include  
include=/opt/SUNWmotif/include  
include=/opt/uimx2.9/include  
include=/usr/openwin/include  
include=$(MOTIFHOME)/include/Xm  
include=$(MAIN_DIR)/include include=$(ACTIFDB_DIR)  
include=$(GIS_DIR)/include  
include=$(API)/map/include include=$(API)/include
```

A.4 Script Genamap

A.4.1 Demo 1 or 2

The scripts are located in the *workarea* of the prototype:
/exe/ZF17/.bootDemo

When calling any of the prototypes, a script is launched to correctly initialize the Demo 1 or 2:

```
# Assume /bin/sh and then find which Bourne Shell is on  
the  
# users PATH.  
#  
global SCRIPTPATH=ZF17:MAP_PATH:$(sys -p -s /bin/sh  
echo \ $GENA/vive/script):\  
$(sys -p -s /bin/sh echo \ $GENA/map_6.2/script)
```

UNCLASSIFIED

A-6

```

global BOURNESHELL=$(sys -p -s /bin/sh which sh)

#
# Check if user's SHELL rc file has output written to
# standard out.
#
if [ -n «$(sys -p exit)» ]
then
    print «${}\nWARNING: Commands in the SHELL rc
    file (e.g. .kshrc or .cshrc) are writing \nto
    standard out. This will affect the operation of
    scripts using 'sys -p'. \nRemove the commands from
    the rc file or redirect their output to
    /dev/tty.${}» >&2
fi

#
# Import UNIX GISRUN value to Genamap. Occasionally
# useful in scripts.
#
global GISRUN=$(sys -p -s $(global BOURNESHELL) echo
\${GISRUN})

#
# Attach the GenaVive service
#
address 10000 probe || attach genavive || exit 1

#
# Attach the Xfer service
#
address 18000 probe || attach xfer || exit 1

#
# Attach the intelligence manager
#
address 55000 probe || attach intManager||exit 1

```

A.4.2 Demo 2

Moreover, Demo 2 uses another script: /exe/ZF17/demo2. When started in method IntManager::SetGenamapWindow, the script performs all necessary steps to «cache», create and make visible the Xfer layers:

```

#
address 10000 probe || attach genavive || exit 1
#
address 18000 probe || attach xfer || exit 1

```

UNCLASSIFIED

A-7

```
#$1 is a parameter passed to demo2 script equal to
#uixid=«$(getv wact)»

uixid=$1

xwindow $uixid W1

drawOverview=«$(xflayer W1 0 0 true Overview)»
draw250k=«$(xflayer W1 2500 1250 true 250k)»
draw50k=«$(xflayer W1 6000 3000 true 50k)»

xfixy Overview 0 0 true
xfixy 250k 0 0 true
xfixy 50k 0 0 true

xfivis Overview true 3
xfivis 250k false 2
xfivis 50k false 1

gvview $drawOverview as Overview $uixid
gvview $draw250k as 250k $uixid
gvview $draw50k as 50k $uixid

gvopen /export/home/auf1/actif/raster/256.tif as map
gvadd map Overview
gvadd map 250k
gvadd map 50k

xfupdate $drawOverview true
gvplot Overview
xfupdate $drawOverview false

xfupdate $draw250k true
gvplot 250k
xfupdate $draw250k false

xfupdate $draw50k true
gvplot 50k
xfupdate $draw50k false
```

A.5 Graphics (Genamap 7.1)

Some scripts were constructed in order to demonstrate the capacity of the Graphic service to meet our requirement. They all refer to the military symbol library located in:

actifPrototype/CommonFeatures/Application/extension/graphics/military.symbols

TABLE A.2: DIRECTORY DESCRIPTION

Directory /exe/ZF17/	Description
arm	Script to draw an armoured symbol over an existing unit.
bc	Script to draw a blue coalition over an existing unit.
bu	Script to draw a blue unit.
demo2.1	Script to initialize symbology testing with graphics, Displayer and military.symbols .
enemy	Script to identify an enemy unit.
friendly	Script to identify a friendly unit.
gu	Script to draw a green unit.
inf	Script to draw an infantry symbol over an existing unit.
init	Script to initialize symbology testing with 'gmplot' functionality.
myzoom	Script to illustrate different scaling using 'zoom', 'viewport' and 'grscale'.
neutral	Script to identify a neutral unit.
rc	Script to draw a red coalition symbol over an existing unit.
refresh	Script to refresh connection to the military symbols library: military.symbols.
ru	Script to draw a red unit.
tactical	Script to initialize tactical symbology layer over an existing map.
wc	Script to draw a white coalition symbol over an existing unit.
yu	Script to draw a yellow unit.

A.6 SQL

Two scripts are available to test Demo 1 with Oracle as shown in Table A.3 below.

TABLE A.3: DIRECTORY DESCRIPTION

Directory	Description
actifPrototype/CommonFeatures/ Application/sql	
create.prototype.table.sql	Create the required table in Oracle and insert six (6) symbols. They are located around the center of the map. Used for a rapid setup of the prototype.
populate.sql	Create the required table in Oracle and insert three hundred (300) symbols. They are located bottom left of the map. It was designed to test the initial display of tactical data.

Type `sqlplus /@actifPrototype/CommonFeatures/Application/
sql/script_name` to run the script.

A.7 Alias & Environment variables

To correctly install Genamap products you must add some paths to your `$(HOME)/.cshrc` file:

```
setenv GENA      /opt/genasys/6.2
setenv GISRUN    $(GENA)/map
setenv WINDOWID 123
setenv API $(GENA)/api

setenv LD_LIBRARY_PATH $(API)/lib
setenv PATH .: $(GENA)/bin:$(GISRUN)/bin
```

To correctly run the prototype you must add some aliases to your `$(HOME)/.actifrc` file:

```
setenv GIS_DIR $(ACTIFHOME)/SituationMap/Gis
```

```
alias intManager      dispatch dispscript -startup
    actifPrototype/CommonFeatures/Appl
ication/src/.bootDemo
```

To correctly enable the .cshrc or .actifrc type source **\$(HOME)/.cshrc**

A.8 Compile

Two prototypes, Demo 1 and Demo 2, were developed in order to demonstrate different approaches when manipulating raster files. Those files are:

TABLE A.4: RASTER FILES INITIALIZATION

Demo 1	Demo 2
/src/Demo1.cpp	/src/Demo2.cpp
/include/Demo1.h	/include/Demo2.h
/uimx/uiIntManager.i	/uimx/uiXferBox.i

In order to compile a specific prototype:

- a) We must rename the Demo X files into:

```
/src/IntManager.cpp
/include/IntManager.h
```

- b) We must then generate the interface code with UIM/X 2.9:

- 1) Select the appropriate «ui*.i» files for Demo X
- 2) Then click on Generate Interfaces code as :
- 3) **actifPrototype/CommonFeatures/Application/src/**
uiIntManager.cpp
- 4) Copy the uiIntManager.h into
actifPrototype/CommonFeatures/Application/include
directory.

To compile any of the prototypes type: **actifPrototype/Make**

A.9 Run

A.9.1 Demo 1

The first prototype demonstrates some GIS functions available via a toolbox integrated into the working situation map view of a «Senior Analyst».

Type `/exe/intManager` to run it.

A.9.2 Demo 2

The second prototype demonstrates the rapid response time when manipulating raster files with the Genasys client service Displayer (Xfer).

Type `/exe/intManager` to run it.

A.10 Font Server

In order to correctly enable the new graphics symbology service, we must start the font server before any access to graphics. The first step is to configure the font path and then add your hostname to it.

Type `$(GENA)/graphics/fonts/configfonts`

Type `$(GENA)/graphics/fonts/xset fp+ tcp/hostname:7200`

ANNEX B – IMPLEMENTATION RULES

B.1 Designing GUI – Creating a GIS window

B.1.1 What?

In Genamap, X-windows is used for graphics display. Multiple GIS windows can be created having a unique alphanumeric name up to 10 characters long. GenaMap provides several commands for managing a GIS window. Therefore, UNIX functions altering X-windows content are not recommended.

B.1.2 When?

In ACTIF, a GIS window should be created each time a user needs a GIS display representation. Thus, the user will be able to move, raise, lower, "iconize" and restore a GIS graphics window. The window is removed when the user no longer displays GIS information.

B.1.3 How?

- a) Create a new window using the 'Generic System Window' widget from the ACTIF palette.
- b) Go into the property editor and change the following properties.

TABLE B.1: PREPRTY EDITOR VALUES

Section	Property	Value
Specific	Background	UxGetBackground(«SYST», «background»);
Specific	DialogTitle	New Title of the window.
Declaration	Name	'ui' + < Domain class name >

- c) Add a «drawingArea» swidget (UIM/X implements MOTIF widget as swidget) into the new window created above.

- d) Go in this form property editor and change the following properties.

TABLE B1.2: PROPERTY EDITOR PROPERTIES

Section	Property	Value	Source
Core	Background	«black»	-
Constraint	BottomAttachment	attach_form	Private
Constraint	LeftAttachment	Attach_form	Private
Constraint	RightAttachment	Attach_form	Private
Constraint	TopAttachment	Attach_form	Private

- e) Under the domain member section class initializing the window, write

```

/*
 * Requesting scripting client from dispatcher
 * architecture where xid is the X-window id.
 */
execute_dispatch_command(SERV_SCRIPT, «sys gxaddcmap -
stdcmap default -id xid»)

```

Note: This sample demonstrates how to add a standard color map to a UIM/X window using the Dispatcher architecture.

B.2 Integrating GIS Components into ACTIF

Connecting to a Genamap Session

B.2.1 What?

In Genamap, a session is a unique type 20 file. A session enables users to establish configuration scenarios, and invokes them many times without re-configuring each time a different device or environment is required. Several other items of information can also be set and retained within a particular session.

B.2.2 When?

In ACTIF, a session should be created before the user accesses GIS modules. Thus, the user will be able to interact directly on a map using a configuration scenario and Genasys functions. The session should terminate when the user no longer accesses GIS modules.

B.2.3 How?

In the startup script of the GIS module, write:

```
#  
# Start up gmap using the appropriate session name  
read session < ./INTERPRO  
attach gmap $session || exit 1  
address 9000
```

In the shutdown script of the GIS module, write:

```
# DESCRIPTION  
# Save the current session environment variables  
# using genv  
#  
genv save
```

B.3 Executing Request broker commands

B.3.1 What?

In Genasys, the dispatcher «Request Broker» architecture facilitates integration of separate software products into one application. The Dispatcher architecture is based on a multi-process message protocol. Each process is started by the Dispatcher.

B.3.2 When?

In ACTIF, a request is sent to the Dispatcher each time a user interaction with the interface requires a GIS response. A request is made up of a service identifier (an integer) and associated message.

B.3.4 How?

Under the domain class member section requesting the dispatcher, write:

```

/*
 * Example of requesting function to the dispatcher.
 */
BOOLEAN execute_dispatch_command
(
    SERV_SCRIPT
    char * command // The command line
)
{
    DISP_ID req;
    int      stat      = 0;
    extern  SERV_ID io[3];
    SERIAL  message;

    // Insert here a command to split the command line into
    // argv
    PackArgs(argc, argv, io, &message);

    req = DispExec( service, &message, NULL);
    if( req < 0 )
    {
        if( CheckDispatcherEvent(0) &&
            GotDispatcherEvent() == DISP_CONN_LOST )
            fprintf( stderr, «\nConnection lost to the
peer client \n» );
        else
            fprintf( stderr, «\n Could not submit
command to peer client \n» );
    }
    else
    {
        while( DispWait(&stat) != req )
            fprintf(stderr, «\nWaiting...\n»);
        if(stat) fprintf( stderr, «\n Client
did not like the command\n»);
    }
    free((void *) message.data);
    free(argv);

    return(TRUE);
};

```

B.4 Converting ADRG

B.4.1 What?

In Genamap, an Arc Digitized Raster Graphics is a unique type 7 file. An import process is conducted in order to read an ADRG CD-ROM and convert it to its GenaMap representation.

B.4.2 When?

In ACTIF, ADRG conversion should be done before any «Senior Analyst» accesses the theater of operation. A system manager collects the map files, then converts and optimizes them to prepare maps for a specific theater of operation.

B.4.3 How?

Under the domain class member section converting ADRG file, write:

```
/*
 * Importing ADRG using scripting client from
 * dispatcher architecture where
 * cdrom_path_to_transh01.thf_directory is the CD-ROM
 * location path of the transh01.thf file.
 */
execute_dispatch_command(SERV_SCRIPT,
    «sys$(GENA)/map_6.2/contrib/bin/adrg2gc -path
    cdrom_path_to_transh01.thf_directory -interpolate -
    colortable - -verbose») ;

/*
 * Optimizing the type 7 file with tiled 256 x 256
 * scheme where file_name.RD is the input file
 * and file_name.tif the output one.
 */
execute_dispatch_command(SERV_SCRIPT, «sys
    $(GENA)/bin/gvconvert -type tiff -format tiled 256
    256 -colortable adrg.CT file_name.RD
    file name.tif») ;
```

B.5 Creating Type 4 Maps

B.5.1 What?

In Genasys, each map or file in a project has a type number associated with it to indicate its content. Some commands only work with or expect particular file types to be used. Vector maps composed of points, lines and areas are recognized by the system as type number 4.

B.5.2 When?

In ACTIF, each time a user creates a new map or saves a new layer, a type 4 is created. Thus, the user will be able to manage the file using Genasys product functions.

B.5.3 How?

Under the domain declaration section class creating the file, write

```
/*
 * Variables
 */
GSDB          *sdb;
GVM_HEAD      *vecthd;
GMAP_HEAD *mapinfo;
GVM_FP        *vectfp;
GMBR_D        mbr;
GPROJECTION   projection;
```

Under the domain member section class creating the file, write

```
/*
 * Variables
 */
if( (sdb= GSDBConnect («Dbpath», «»,«»))==NULL)
{
    fprintf(stderr, «Error connecting database \n»);
    return(0);
}
```

Under the domain member section class creating the file, create a "mbr":

UNCLASSIFIED

B-7

```

/*
 * Set the mbr values. Ex:
 */
mbr.xmin = -124.0015869;
mbr.ymin = 47.9999695;
mbr.xmax = -120.0010986;
mbr.ymax = 50.0000992;
mbr.zmin = mbr.zmax = 0.0;

```

Under the domain member section class creating the file, choose the projection:

```

/*
 * Set the projection value. Ex:
 */
projection.code = GCTP_SYS_GEOGRAPHIC;
projection.zone = 0;
projection.units = GCTP_DEGREES;
projection.ellipse = GCTP_ELL_CLARK1866;

```

Under the domain member section class creating the file, create a generic map header:

```

/*
 * Create a map header object. Ex:
 */
mapinfo = GInstantiateMapHeader(
    «cotel»,
    GVECTOR_2D,
    «Vector map representing the initial
    display»,
    250000,
    GTHIRTY_TWO_BIT,
    &mbr,
    &projection
);

```

Under the domain member section class creating the file, create a vector map header:

```

/*
 * Create a vector map header object.
 */
if( (vecthd = GVMInstHeader( mapinfo))==(GVM_HEAD*)0)
{
    fprintf(stderr, «Error (%d): %s\n», GErrorNum(),

```

```

GErrorMess() );
return(0);
}
// Free the mapinfo
GFreeMapHeader(mapinfo);

```

Under the domain member
vector map:

```

/*
 * Create a new vector
 * the physical id
 */
if( !GVMCreate(sdb,
{
fprintf(stderr, «Error:
GErrorMess() );
return(0);
}

```

UNCLASSIFIED

B-8

fo variable
info);

number section class creating the file, create a new

vector map where map_name represents
identification of the map.

, «map_name», vecthd))

Error (%d): %s\n», GErrorNum(),

ANS CLASSIFICATION

UNCLASSIFIED

C-1

ANNEX C – PERFORMANCE & STATISTICS

C.1 Testing Environment

Some analyses were carried out on the two prototypes Demo 1 and Demo 2 to produce system activity reports. The UNIX program 'sar' [19] was used to collect processor, disk and memory activity each 5 seconds. Refer to the UNIX 'man' page for detailed information on this command.

These tests were made under the following environment:

- Sparc station 5

- Microsparc 85 MHz

- Video card Turbo GX

- 64 Mb RAM

- Internal disk (SD3) 1.05 Gb

- External disk (SD2) 2.1 Gb

- 128 Mb swap space located on SD3

- Solaris 2.5

- Application and User account stored on SD2

Each activity reading was based on a graphical user interface interacting with a type 7 (raster) file of 25 Mb or a type 4 (vector) file of 17k. Actions have been conducted following this schedule:

SANS CLASSIFICATION

TABLE C.1: DEMO 1 EXPERIMENTATION SCHEDULE

Time	Action
11:29:00	The Demo 1 is launched from the actifPrototype/CommonFeatures/Application/exe/ intManager.
11:29:30	«Area of Operation Views...» is selected from menu bar Sitemap. The working situation map view is displayed.
11:30:30	The toolbox function «distance» is called by clicking «Distance».
11:31:00	The toolbox function «zoom in» is called by clicking «Zoom +».
11:31:10	The toolbox function «zoom in» is called by clicking «Zoom +».
11:32:00	The toolbox function «zoom out» is called by clicking «Zoom -».
11:32:10	The toolbox function «zoom out» is called by clicking «Zoom -».
11:32:30	The toolbox function «zoom out» is called by clicking «Zoom -».
11:33:10	The toolbox function «zoom in» is called by clicking «Zoom +».
11:34:00	The «panning» function is called by clicking «Pan».
11:34:30	The «querying object» function is called by clicking «? Obj».
11:35:30	The «panning» function is called by clicking «Pan».
11:36:00	The application terminates.

TABLE C.2: DEMO 2 EXPERIMENTATION SCHEDULE

Time	Action
13:10:30	The Demo 2 is launched from the actifPrototype/CommonFeatures/Application/exe/ intManager.
13:18:34	The caching of the images terminates and the toolbox is displayed
13:19:00	The toolbox radio buttons are clicked continuously to change the current view scale.
13:19:30	The radio button test is stopped.
13:20:00	The toolbox arrows are selected to scroll the overview level.
13:20:10	The arrow button test is stopped.
13:20:30	The toolbox arrows are selected to scroll the middle view level.
13:20:38	The arrow button test is stopped.
13:21:00	The toolbox arrows are selected to scroll the nearest view level.
13:21:08	The arrow button test is stopped.
13:21:30	The application terminates.

C.2 Terminology Definitions

Runnable Queue: 'runocc'

The 'runnable' queue is the primary measure that can be used to decide whether a system needs more CPU power. If a process is ready to run but there is no free CPU to run on, then the process waits in this queue. On a multiprocessor machine, the queue empties more quickly than on a unique processor. A CPU time slice is about 50 to 100 ms, and a service time is up to four time-slices is usually perceived to be acceptable performance. 'runocc' is the proportion of the time that there is something in the run queue [19].

Swap Space: 'freeswap'

'freeswap' shows the free swap in 512-byte blocks. 'freeswap' is equal to available swap + reserved swap [19].

Free Memory: 'freemem'

'freemem' reports memory in pages of RAM that are immediately ready to be used whenever a process starts up or needs more memory [19].

Minor Faults: 'vflt'

A minor fault is caused by an address space or hardware address translation fault [19].

Other Fault Types: 'pflt'

Protection fault are caused by illegal access of the kind that produce "segmentation violation - core dumped" messages [19].

Pages Paged In: 'pgin', 'ppgin'

The number of page faults and the number of pages paged in by swap space or file system reads. Since the file system block size is 8 Kbytes, there are often two pages or 8 Kbytes, paged in per page fault [19].

Pages Paged Out: 'pgout', 'ppgout'

The number of page-outs and the number of pages paged out to the swap space or the file system. Because of the clustering that occurs on swap space writes, there may be very many pages written per page-out [19].

Pages Freed: 'pgfree'

Pages freed is the rate at which memory is being put onto the free list by the page scanner daemon. The rate is in pages freed per second [19].

The Page Daemon Scanning Rate: 'pgscan'

These commands report the number of pages scanned by the page daemon as it looks for pages to steal from processes that are not using them often. This is the key memory shortage indicator if it stays above 200 pages per second for long periods [19].

The Activity data reported: 'avserv', 'await', 'avque'

'AVSERV' Average time, in milliseconds, for a transfer request to be completed by the device (for disks, this include seek, rotational latency and data transfer times).

'AWAIT' Average time, in milliseconds, that transfer requests wait idly in the queue (measured only when the queue is occupied).

'AVQUE' The sum of the average wait time plus the average service time.

C.3 Test Results

The following analyses are followed by summaries and graphic reports on each activity.

C.3.1 Demo 1

The first demo was developed around the «IntManager» graphical user interface prototype. The biggest memory need was at the initialization of the

application and the GIS engine: around 250 Kb (Figure C.1). The application kept an average need of 164 Kb throughout. The maximum swap space used was 11.7 Mb and the swap space needed kept an average size of 10.1 Mb (Figure C.2). The processor activity for Demo 1 tells us that 73 percent of the time there was something in the run queue. The results tell much about the approach for Demo 1. It shows that each action (e.g. «Zoom -», «Pan») needs to free space (ppgout/s, pgout/s) and load (ppgin/s, pgin/s) the required data in memory. This approach may need some processing time when the system searches for memory page (pgscan/s), resolves faults (vflt/s, pflt/s) and allocates pages in the swapping area (blks/s(internal)).

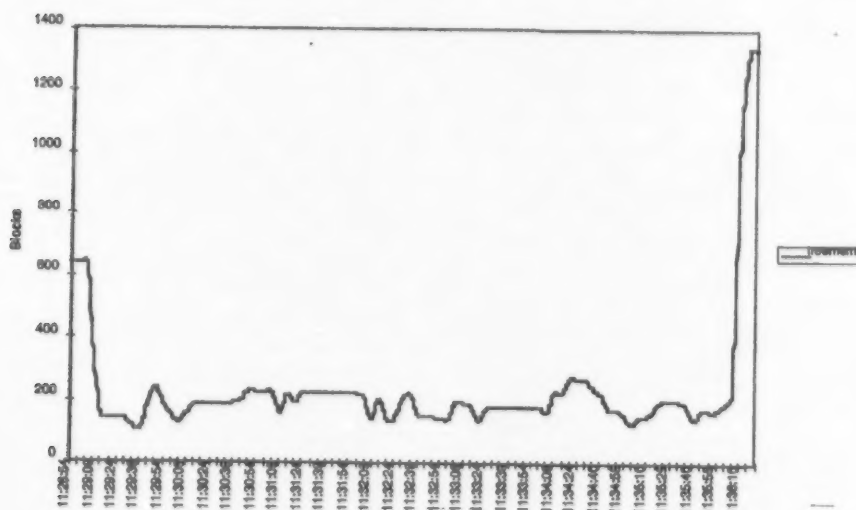


FIGURE C.1- Demo 1 Free Memory

'freemem' reports memory in pages of RAM that are immediately ready to be used whenever a process starts up or needs more memory. The biggest memory need was at the initialization of the application and the GIS engine between 11:28:54 and 11:29:30 (around 250Kb). The application kept an average need of 164Kb throughout.

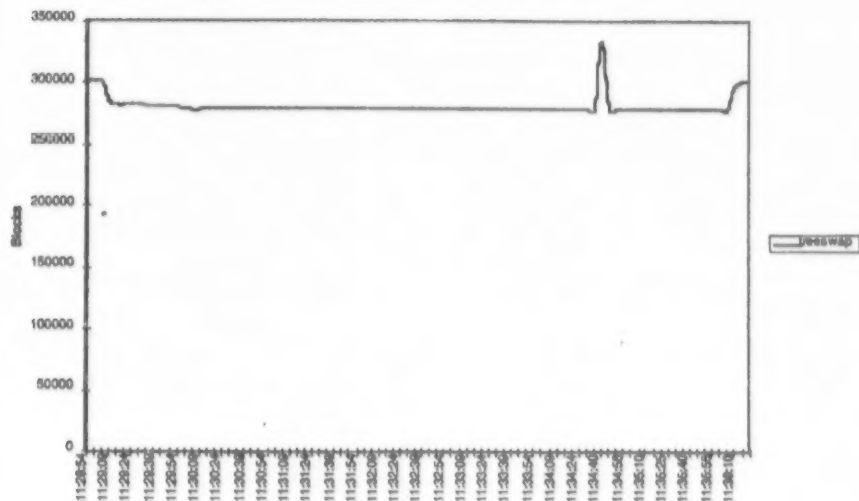


FIGURE C.2- Demo 1 Free Swap

'freeswap' shows the free swap in 512-byte blocks. 'freeswap' is equal to available swap + reserved swap. The maximum swap space used was 11.7 Mb and the swap space needed kept an average size of 10.1 Mb.

C.3.2 Demo 2

The second demo uses a standalone graphical user interface. The biggest memory need was at the initialization: around 3.7 Mb (Figure C.3). The application kept an average memory need of ~3 Mb throughout. The idea of Demo 2's approach was to «cache» the same image as Demo 1 in three distinct scales. Thus reserved swap space was divided in three layers: overview (31.9 Mb), 1: 250 000(2.9 Mb) and 1: 50 000(17.2 Mb) for a maximum of ~53 Mb (Figure C.4). The swap area needed for Demo 2 had an average size of 42 Mb. The processor activity for Demo 2 tells us that 79 percent of the time there was something in the run queue. The results show that each action (e.g. «Zoom -», «Pan») was very rapid because it involved data already loaded into memory. On the other hand, this approach needed some time (several minutes) to initialize: the system read the input data (blks/s(external)),

searched for memory pages (pgscan/s), resolved faults (vflt/s, pflt/s), allocated memory pages (ppgin/s, pgin/s) and stored them in the swapping area (blks/s(internal)).

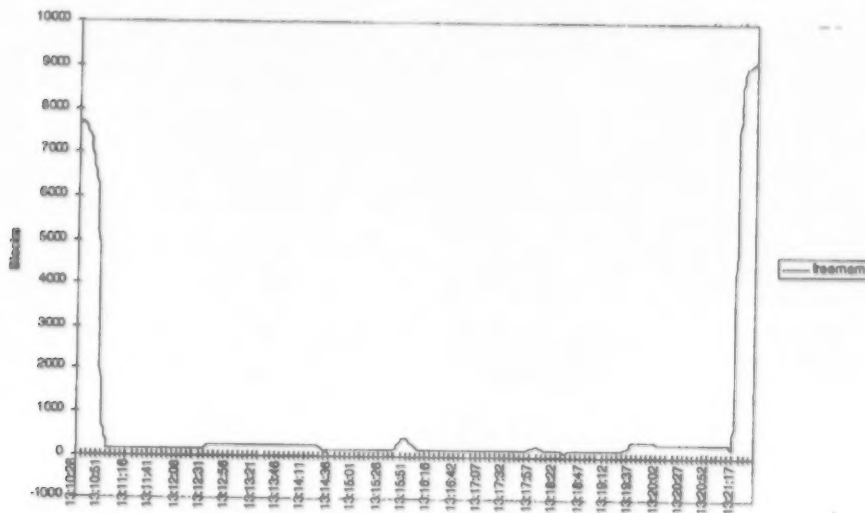


FIGURE C.4- Demo 2 Free Memory

'freemem' reports memory in pages of RAM that are immediately ready to be used whenever a process starts up or needs more memory. The biggest memory need was at the initialization of the application and the "caching" process between 13:10:26 and 13:18:34 (around 3.7 Mb). The application kept an average need of ~3 Mb throughout.

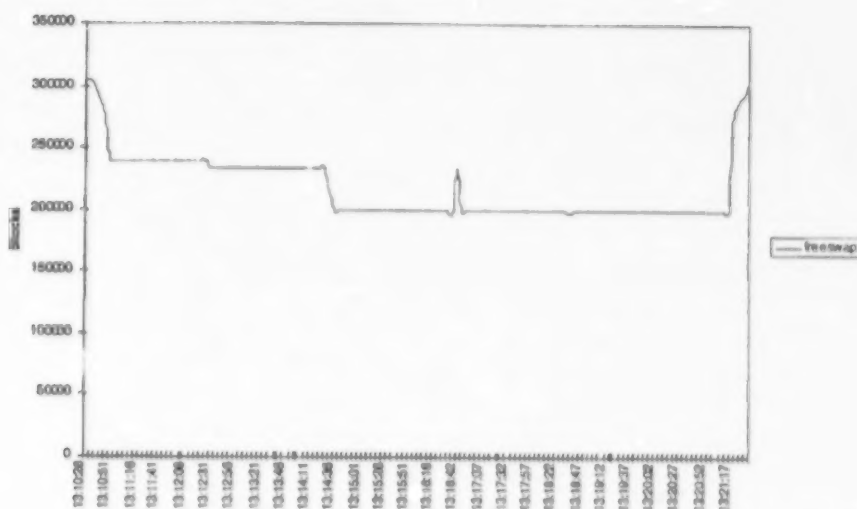


FIGURE C.5- Demo 2 Free Swap

'freeswap' shows the free swap in 512-byte blocks. 'freeswap' is equal to available swap + reserved swap. Reserved swap space was divided in three layers: overview(31.9 Mb), 1:250000(2.9 Mb) and 1:50000(17.2 Mb) for a maximum of ~53 Mb. The swap area needed for Demo2 had an average size of 42 Mb.

C.4 Result Summary for Part I

C.4.1 Processor activity average

TABLE C.3: DEMO 1

%usr	%sys	%wio	%idle	%runocc
50	45	0	6	73

TABLE C.4: DEMO 2

%usr	%sys	%wio	%idle	%runocc
58	36	1	4	79

C.4.2 Disk Activity Average

TABLE C.5: DEMO 1

device	blks/s	r+w/s	avserv ms	avwait ms	avque ms
SD2 (external disk)	11	1	10.5	0	0
SD3 (internal disk)	14	0	13	0	0

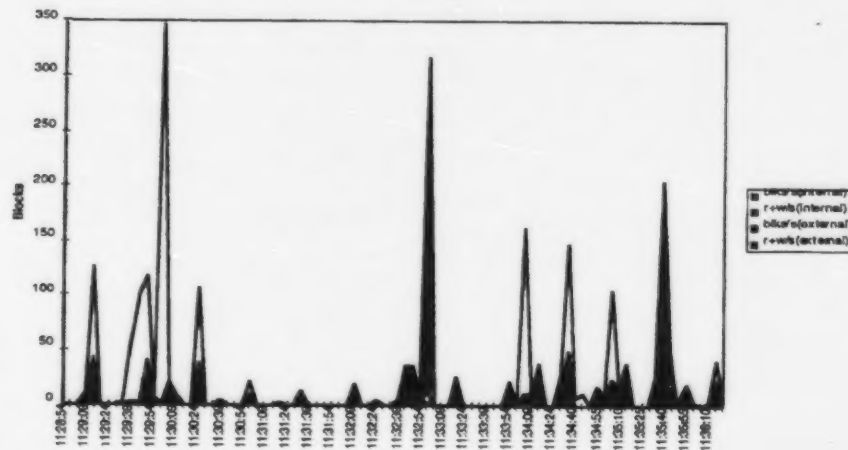
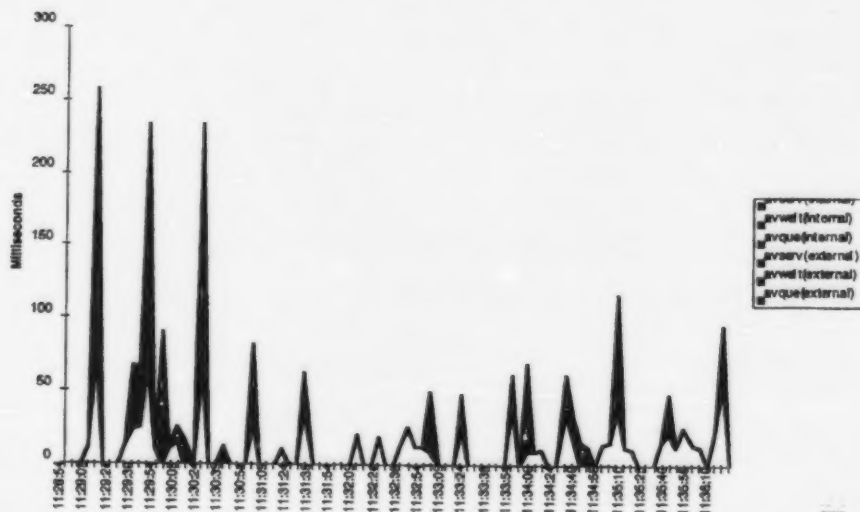


FIGURE C.5- Demo 1 Disk I/O



UNCLASSIFIED

C-10

FIGURE C.6- Demo 1 Disk Activity
TABLE C.6: DEMO 2

device	blks/s	r+w/s	avserv ms	avwait ms	avque ms
SD2 (external disk)	166	4	11.4	0	0.1
SD3 (internal disk)	91	4	13	0	0.1

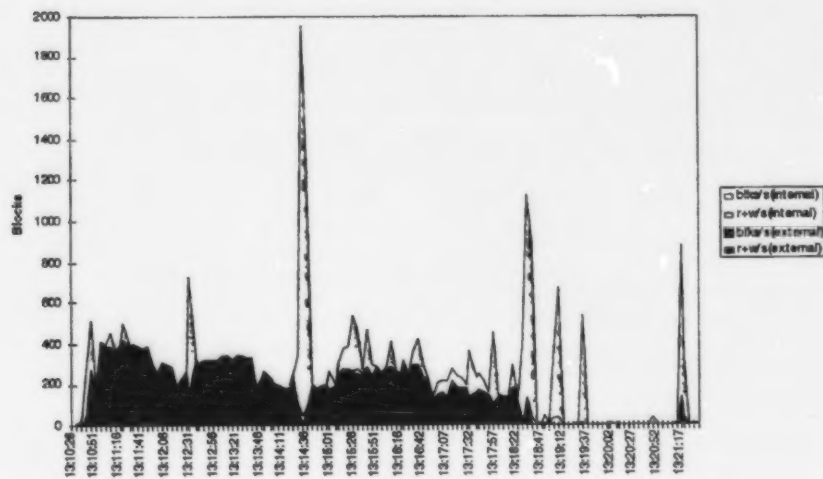


FIGURE C.7- Demo 2 Disk I/O

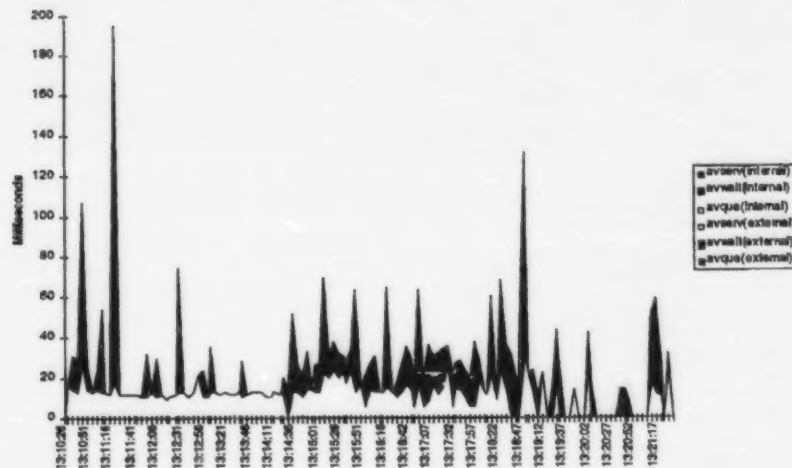


FIGURE C.8- Demo 2 Disk Activity

SANS CLASSIFICATION

C.5 Memory Activity Average

TABLE C.7: DEMO 1

pgout/s	pgin/s	ppgout/s	ppgin/s	vflt/s	pflt/s	pgscan/s	pgfree/s	freemem ¹	freeswap ²
0.07	0.6	1.17	1.2	19.48	8.17	19.24	1.81	240	281076

¹ Reports memory in pages of RAM

² Reports free swap in 512-byte blocks

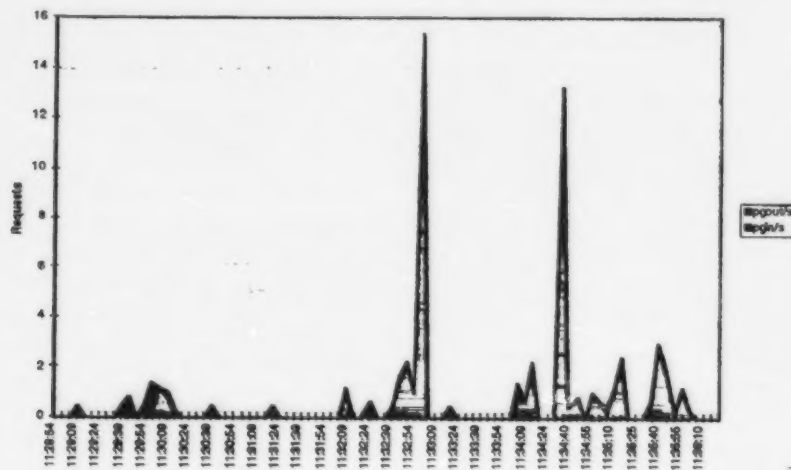


FIGURE C.9- Demo 1 Memory Request Activity

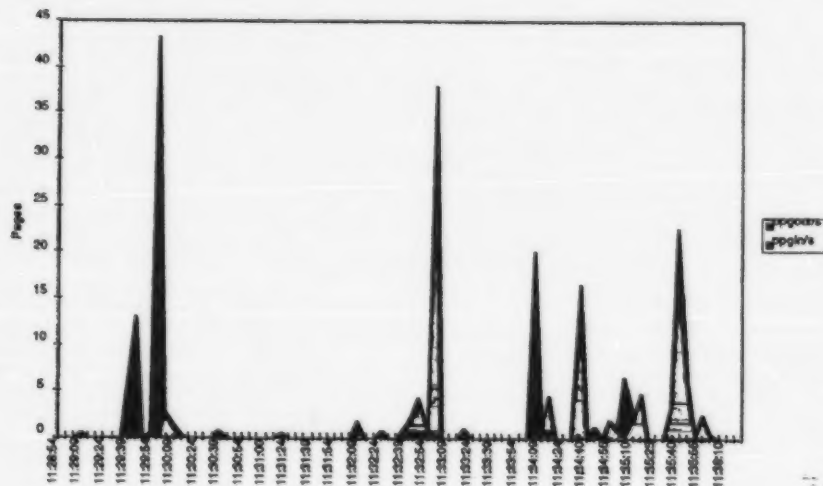


FIGURE C.10- Demo 1 Paging Activity

UNCLASSIFIED

C-12

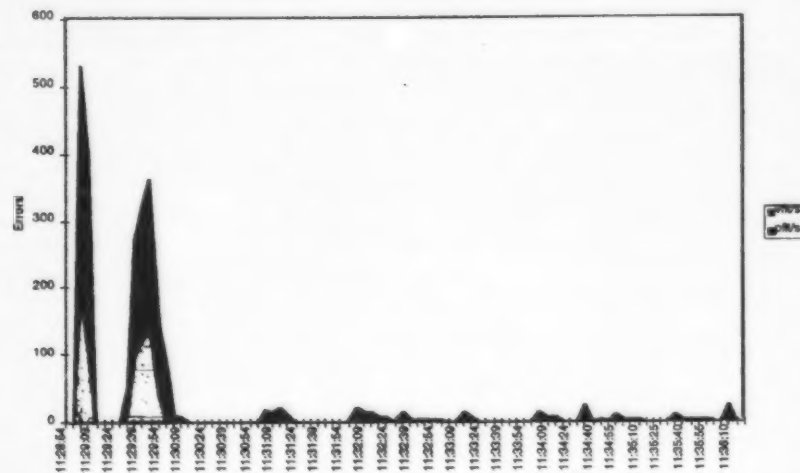


FIGURE C.11- Demo 1 Page Faults

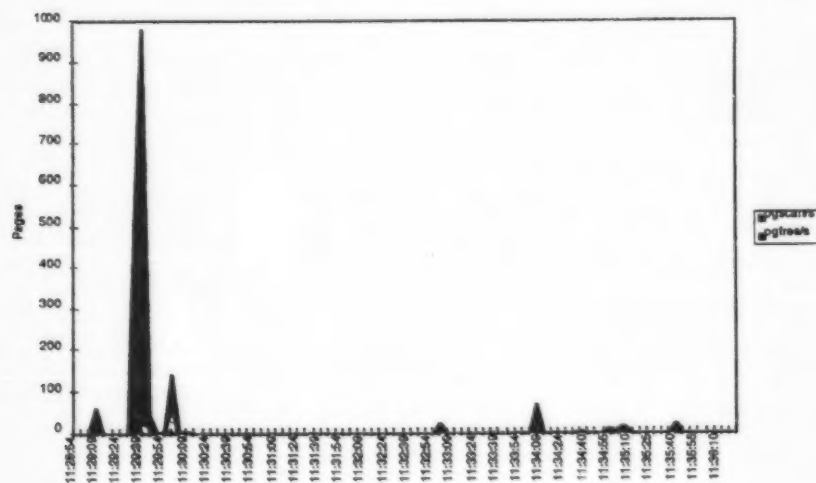


FIGURE C.12- Demo 1 Page Stealing Activity

SANS CLASSIFICATION

TABLE C.8: DEMO 2

pgout/s	pgin/s	ppgout/s	ppgin/s	vflt/s	pflt/s	pgscan/s	pgfree/s	freemem ¹	freeswap ²
0.25	7.93	7.01	25.67	26.6	3.58	47.99	14.64	718	217665

¹ Reports memory in pages of RAM

² Reports free swap in 512-byte blocks

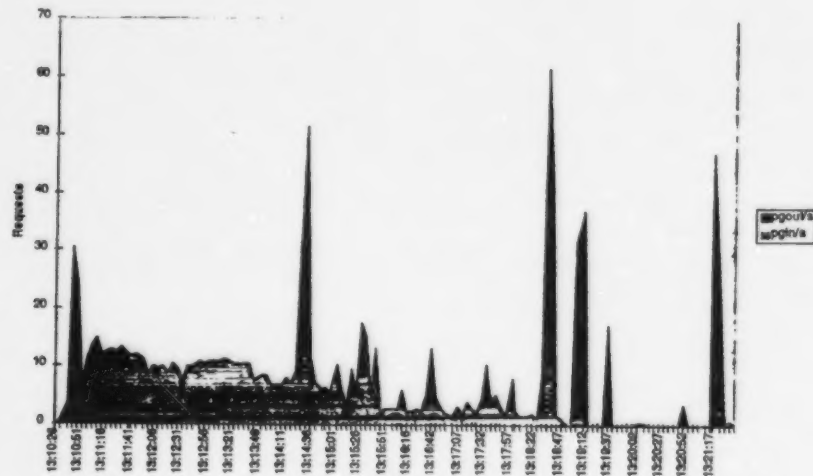


FIGURE C.13- Demo 2 Memory Request Activity

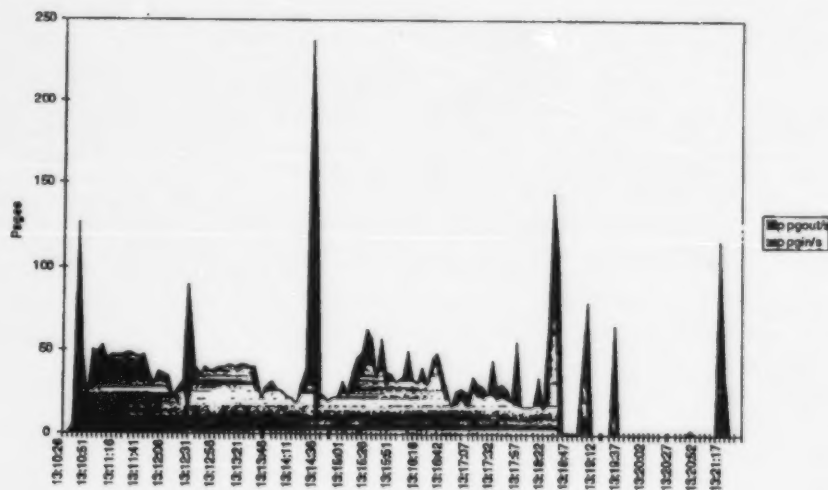


FIGURE C.14- Demo 2 Paging Activity

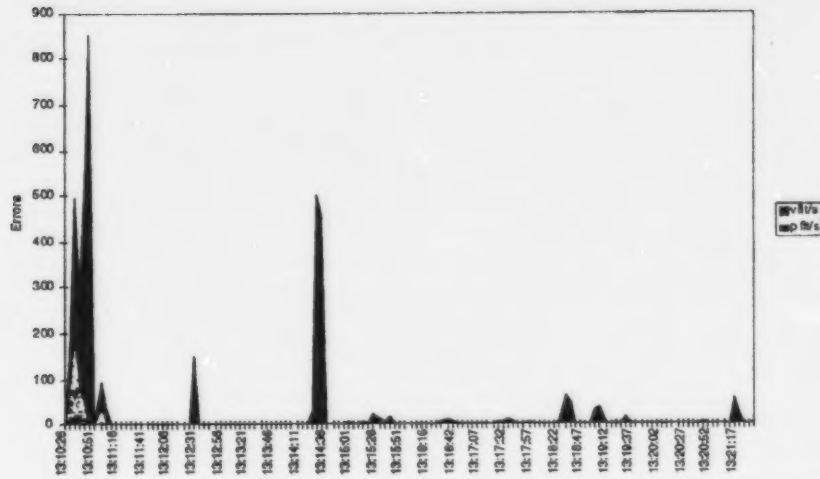


FIGURE C.15- Demo 2 Page Faults

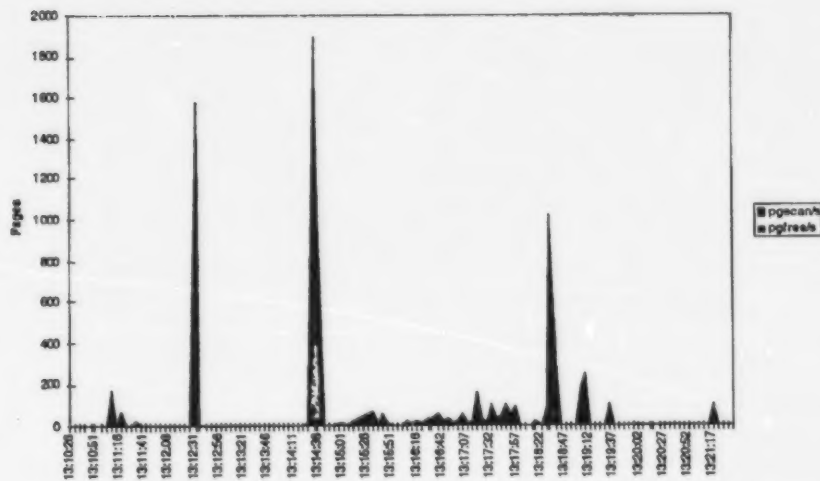


FIGURE C.16- Demo 2 Page Stealing Activity

C.6 Result Summary for Part II

C.6.1 Import of ADRG files from the Geomatics Canada CD-ROM

An ADRG file xxca0103.img of ~213 Mb was converted into a type 7 file of ~92 Mb called xxca0103.RD in approximately 45 minutes on a Sparc station 5.

A type 7 file xxca0103.RD of ~93 Mb was Optimized into a file of ~24 Mb called xxca0103.tif in approximately 5 minutes on a Sparc station 5.

C.6.2 GIS Toolbox

Depending on the map scale, the performance of the panning function varies as shown in Table C.8 below. Using Displayer, each click on an arrow button produces an instant refresh of the «virtual window» over the «cache».

TABLE C.9: REFRESH TIMES OVER MAP SCALES

Scale	Time to refresh
1:1000000	5 to 35 seconds
1:500000	6 to 12 seconds
1:250000	6 to 8 seconds
1:100000	5 to 6 seconds

Depending on the map scale, the performance of the zooming function varies as shown in Table C.9. Using Displayer, each click on a radio button produces an instant refresh of the «virtual window» over the «cache».

TABLE C.10: REFRESH TIMES AFTER ZOOMING

	Time to	Refresh
Scale	Zoom -	Zoom +
1:1000000	~23 seconds	
1:500000	~8 seconds	~7 seconds
1:250000	~6 seconds	~6 seconds
1:100000		~5 seconds

C.6 Initial display of tactical data

C.6.1 Demo 1

The total initial display time was ~22 seconds on a Sparc station 5. We used two Oracle tables: OBJECT and POSITION containing 300 records each. The Oracle connection to SQL*net was established at the beginning of the «IntManager» prototype. The «CreateTactical» method adds 2 to 3 seconds to the total initial display time.

C.6.2 Demo 2

The total initial time to «cache» our three maps, each at a different scale, was 8 minutes on a Sparc station 5 (3 minutes on a Sparc Ultra 140).